HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

Diplomarbeit

Herr Ing.
Michael Erwin Hödl

Schnittstelle, Steuerung und GUI für ein LKW-Kolonnenparksystem

Mittweida, 2018

Angewandte Computer- und Biowissenschaften

Diplomarbeit

Schnittstelle, Steuerung und GUI für ein LKW-Kolonnenparksystem

Autor:
Herr Ing. Michael Hödl

Studiengang:
Technische Informatik

Seminargruppe:
KT14wEA

Erstprüfer:
Prof. Dr.-Ing. Swen Schmeißer

Zweitprüfer:
Dipl.-Ing. Berndhard Jagersbacher

Einreichung:
Mittweida, 20.08.2018

Verteidigung/Bewertung:
Mittweida, 2018

Bibliografische Beschreibung:

Hödl, Michael Erwin:
Schnittstelle, Steuerung und GUI für ein LKW-Kolonnenparksystem
59 Seiten, 28 Abbildungen, 2 Anlagen mit Systemaufbau und GUI,
Hochschule Mittweida (FH), Fachbereich Computer- und Biowissenschaften

Diplomarbeit, 2018

Referat:

Die vorliegende Diplomarbeit befasst sich mit der Programmierung einer Schnittstelle, Steuerung und Visualisierung für ein intelligentes LKW-Kolonnenparksystem. Das intelligente LKW-Kolonnenparksystem soll die Verkehrsteilnehmer über freie Parkplätze oder Stellplätze informieren.
Das Hauptziel ist, alle Module auf dem Parkplatz in ein übergeordnetes Steuerungs- und Visualisierungssystem zu integrieren.
Bei dieser Arbeit handelt es sich um eine Neuentwicklung mit einem vordefinierten Produkt, deshalb ist für eine qualifizierte Programmierung, die Kenntnis des Produkts und seiner Schnittstellen erforderlich.

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AG	Auftraggeber
API	Application Programmable Interface
AX	Automation X
DA	Digitaler Ausgang
DC	Direct Current (Gleichstrom)
DCS	Prozessleitsystem(Distributed Control System)
DE	Digitaler Eingang
FUP	Funktionsplan
HD	High Definition
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
LED	Leuchtdiode
LIDAR	Light Detection and Ranging
IOT	Internet of Things
JSON	JavaScript Object Notation
PSR	Parkstandsreihe
SCADA	Betriebsleitsystem (Supervisory Control and Data Acquisition)
Soft SPS	Speicherprogrammierbare Steuerung in Software
MS SQL	Microsoft Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Technische Lieferbedingungen für Streckenstationen
T SQL	Transact Structured Query Language
REST	Representational State Transfer
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

0 Einleitung

Im einleitenden Kapitel werden die Motivation und die Aufgabenstellung dieser Diplomarbeit besprochen. Gleichzeitig erfolgt ein kurzer Überblick der einzelnen Kapitel dieser Arbeit.

0.1 Motivation

Das System eines intelligenten LKW-Kolonnenparksystem wird von der südbayerischen Autobahndirektion gefördert und in Kiefersfelden, nahe der österreichischen Grenze, erstmalig eingesetzt und getestet, um Information und mögliche Parkraumersparnisse zu gewinnen.

Die vorliegende Arbeit befasst sich im Rahmen der Aufgabenstellung mit der Programmierung einer Schnittstelle, Steuerung und GUI zu dem intelligenten Kolonnenparksystem der Firma Neurosoft.

Das Hauptziel ist es dem Kunden eine gesamte Lösung zu präsentieren, in der das intelligente LKW-Kolonnenparksystem mit einer offenen Schnittstelle in ein Steuerungs- und Visualisierungssystem zu integrieren ist, um von den zentralen Servern aus den gesamten Parkplatz zu steuern und den Autobahndirektionsmitarbeitern eine übersichtliche Visualisierung zur Verfügung zu stellen.

Des Weiteren sollen möglichst offene Industriestandards oder ein Webservice für die Maschine Zu Maschinen-Kommunikation für die Integration in das Steuerungs- und Visualisierungssystem verwendet werden. So ist mit geringeren Entwicklungs- und Abklärungskosten zu rechnen ist.

Für den AG muss die endgültige Visualisierung und Parametrierung des intelligenten LKW-Kolonnenparksystems mit einem geringen Aufwand zu erstellen sein. So kann dieses Verfahren und das am Ende des Projekts entstandene Produkt möglicherweise als Gesamtprodukt auf anderen Autobahnrastplätzen eingesetzt werden.

0.2 Kapitelübersicht

Die Diplomarbeit besteht aus 5 Kapiteln.

Nach der allgemeinen Einleitung des **Kapitel 1** werden alle notwendigen und geforderten Funktionen des Steuerungs- und Visualisierungssystem in **Kapitel 2** erläutert.

In **Kapitel 3** wird das Systemkonzept für das intelligente LKW-Kolonnenparksystem erläutert. Mit diesem Konzept wird dann das Steuerungs- und Visualisierungssystem von der Firma AutomationX programmiert und designt.

Anschließend wird in **Kapitel 4** die komplette Umsetzung der gewünschten Anforderungen des Kunden mit den definierten Schnittstellen implementiert. Nachdem die Schnittstellen implementiert wurden, widmet sich die Arbeit zuerst dem Thema Anbindungen von Schnittstellen, der automatische Steuerungsabläufe und danach der Visualisierung.

Schließlich werden in **Kapitel 5** noch einmal die Resultate der einzelnen Kapitel der Diplomarbeit bewertet. Hier werden die wichtigsten Punkte zusammengefasst. Nach Abschluss des Projekts folgt ein Resümee zur umgesetzten Lösung und eine abschließende Bewertung möglicher Alternativen. Des Weiteren gibt es einen kleinen Ausblick, um welche Funktionen das bestehende System erweitert werden kann, damit der Betrieb eines LKW-Kolonnenparksystems noch komfortabler wird.

1 Grundlagen und Stand der Technik

In diesem Kapitel werden die Grundlagen der einzelnen Systemkomponenten, sowie der Datenaustausch bis hin zur Steuerung dargestellt. Es beinhaltet einen Überblick über die einzelnen Komponenten des intelligenten LKW-Kolonnenparksystem, eine Erläuterung von relevanten Funktionen eines modernen Steuerungs- und Visualisierungssystems und Möglichkeiten von Zusammenspiel und Datenaustausch dieser Systeme.

1.1 Intelligentes LKW-Kolonnenparksystem

Beim intelligenten LKW-Kolonnenparken handelt es sich um eine Initiative und ein Pilotprojekt des Bundesministeriums für Verkehr und digitale Infrastruktur.

Nach der Homepage des Bundesministeriums für Verkehr und digitale Infrastruktur lautet die Definition für das intelligente LKW-Kolonnenparken aus der Quelle [bmvi2018] wie folgt:

„Beim telematisch gesteuerten Kolonnenparken werden die Lkw-Fahrer mittels einer Schranke an einem Terminal angehalten, um ihre Abfahrtszeit einzugeben. Sie bekommen dann vom System eine Parkreihe zugewiesen. Im Ergebnis parken mehrere Lkw entsprechend ihrer Abfahrtszeit hintereinander. Hierdurch kann eine Ausfahrspur entfallen und somit die Nutzung vorhandener bzw. geplanter Fläche optimiert werden. Zu diesem besonderen Parkverfahren wurde auf der Rastanlage Kiefersfelden an der A 93 in Südbayern ein Projekt durchgeführt. „

Für das Projekt intelligentes LKW-Kolonnenparken sind bereits einzelne Anlagenteile auf dem Parkplatz montiert. Diese bilden ein in sich geschlossenes System. Die übergeordnete Steuerung und Visualisierung wurde noch nicht realisiert und muss erst mit den Herstellern der einzelnen Anlagenteile definiert werden, weil es sich um eine Soft- und Hardware Neuentwicklung handelt. Das heißt, dass bei den meisten Anlagenteilen das Interface zur Kommunikation oder auch noch die Entscheidung, welche Daten und Funktionen zur Verfügung gestellt werden sollen, erst mit den jeweiligen Herstellern definiert und entwickelt werden.

In den folgenden Unterkapiteln werden die einzelnen Systemkomponenten des intelligenten LKW-Kolonnenparkens beschrieben.

1.1.1 Kolonnenparken-Kontroller

Der Kolonnenparken-Kontroller dient der kompletten Erfassung aller Fahrzeuge, die in den Parkplatz einfahren. Mit den gesammelten Daten sollte dann ein komplettes Abbild

des Parkplatzes und der einzelnen Parkstandsreihen in einer Datenbank abgebildet und wenn möglich, einer übergeordneten Steuerung zur Verfügung gestellt werden. Als erweiterte Funktion sollen Attribute verändert werden können.

Der Kolonnenparken-Kontroller ist in drei Einzelkomponenten aufgeteilt, die in den nachfolgenden Unterkapiteln näher beschrieben werden.

1.1.1.1 Fahrzeugerkennung

Im Einfahrtsbereich des Parkplatzes wird das Fahrzeug von hochauflösenden Kameras detektiert und anhand einer mitgelieferten Videoerkennungssoftware ausgewertet. Die Daten, die die Videoerkennungssoftware erkennt, werden in einer Datenbank gespeichert. Folgende Merkmale können erkannt werden:

- Kennzeichen
- Fahrzeugtyp
    - PKW
    - LKW
    - Sattelschlepper
    - LKW mit Anhänger
    - …
- Länge
- Höhe
- Breite

Mit diesen erfassten Merkmalen wird das von der Videoerkennungssoftware erfasste Fahrzeug auf dem ganzen Parkplatz mit den anderen Unterkomponenten wiedererkannt. Eine lückenlose Verfolgbarkeit des Fahrzeuges ist das Resultat.

1.1.1.2 Restlängenermittlung

Der Parkplatz besteht aus 31 Parkplatzreihen, die auf jeder Parkstandsreihen-Einfahrt mit einem 2D-LIDAR Sensor[1] ausgestattet sind, um den freien Parkbereich in der jeweiligen Reihe zu erkennen. Auf dem Markt befinden sich bereits 3D LIDAR Sensoren, um eine bessere Qualität für eine Restlängenermittlung durchzuführen, aber der Berechnungs- und Entwicklungsaufwand erhöht sich dadurch enorm. Die Restlängenermittlung liefert dem Kolonnenparken-Kontroller die nötige Information, ob in der ausgewählten Parkstandsreihe noch genügend Platz für ein weiteres Fahrzeug zur Verfügung steht.

[1] LIDAR Sensor dienen der optischen Abtastung von Objekten (siehe https://de.wikipedia.org/wiki/Lidar)

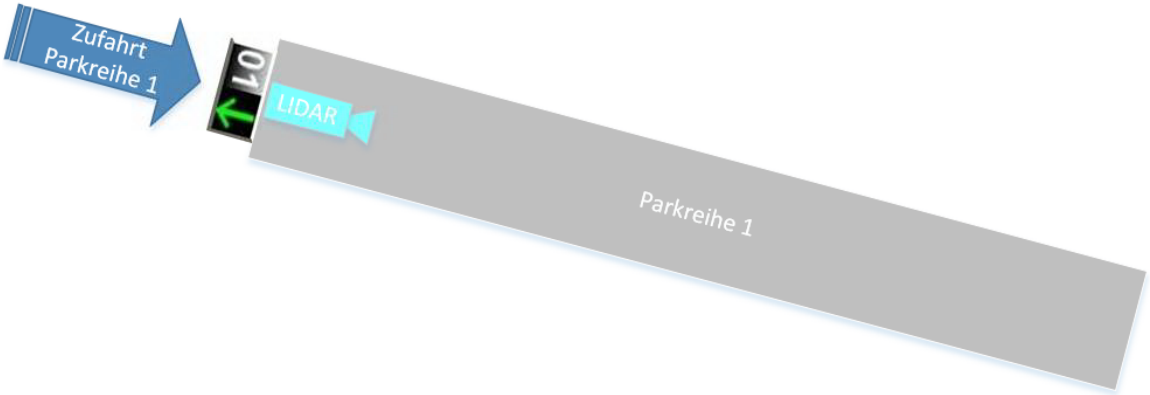


Abbildung 1: Übersicht Parkstandsreihe

Folgende Werte werden vom 2D LIDAR Sensor erkannt:

- Gesamte Länge der Parkstandsreihe
- Freie Länge der Parkstandsreihe
- Störung des Sensors

1.1.1.3 Fahrzeugverfolgung

Um die genaue örtliche Position eines eingefahrenen Fahrzeugs zu bestimmen, ist eine Fahrzeugverfolgung am ganzen Areal notwendig. Zusätzlich muss auch eine genaue Fahrzeugzuordnung funktionieren um gegebenenfalls Falschparker oder die am Einfahrtsbereich eingegebene Abfahrtszeit zu erkennen. Diese Fahrzeugverfolgung wird auch mit dem 2D LIDAR System durchgeführt. Das LIDAR System vergleicht die Merkmale mit der Fahrzeugerkennung am Einfahrtsbereich. Um diese Fahrzeugverfolgung zu gewährleisten wurden hier nicht nur die einzelnen Parkstandsreihen mit den Sensoren ausgestatten, sondern auch die einzelnen Zufahrts- und Ausfahrtsbereiche. Damit eine Zuordnung der Fahrzeugerkennung erleichtert wird, wurde der Parkplatz in einzelne Bereiche unterteilt.

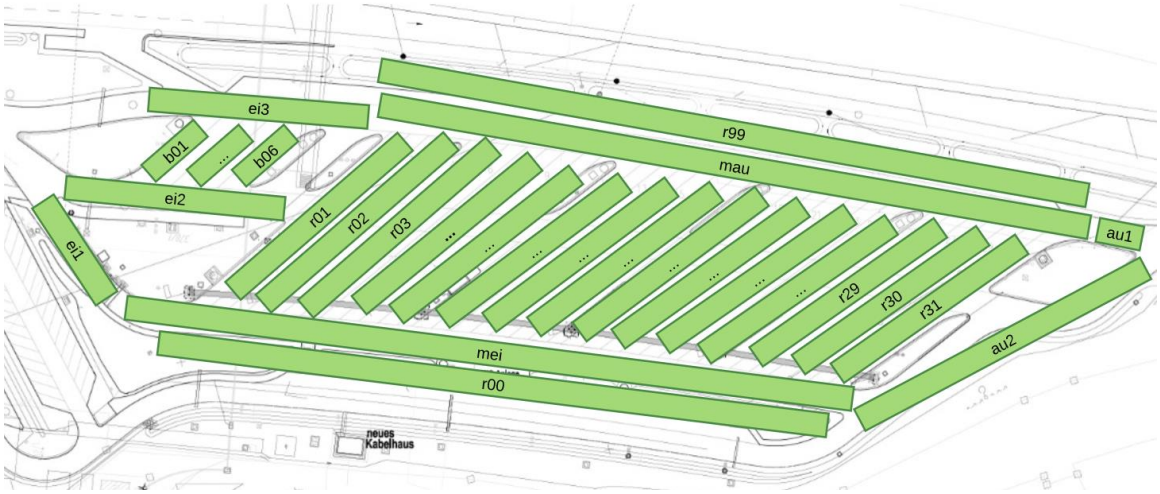


Abbildung 2: Aufbau des Parkplatzes aus der Quelle [Aufbau2018]

Folgende Information werden von den einzelnen Sensoren erfasst:

- Genaue Position im jeweiligen definierten Bereich
- Fahrzeug ID
- Störung des Sensors

1.1.2 Einfahrtsbereich

Damit nicht jeder unangemeldet in den Parkplatz einfahren kann, ist hier wie in Parkgaragen ein Terminal mit zwei Touch-Monitoren verbaut. Danach befinden zwei Schranken, die über eine übergeordnete Steuerung geöffnet oder geschlossen werden können.

In den folgenden Unterkapiteln werden die einzelnen Systemkomponenten des Einfahrtsbereichs beschrieben.

1.1.2.1 Terminal

Das Terminal hat 2 Ausbuchtungen für Touch-Monitore und 3 Ticketdrucker. Die Touch-Monitore dienen der Eingabe der gewünschten Abfahrtszeit der Fahrer und sind auf 2 verschiedenen Höhen montiert. Das erleichtert den Zugang der Fahrer aus unterschiedlichen Fahrzeughöhen.

Im Innenraum des Terminals befinden sich Freibereiche um Industrie PCs, Netzwerkswitche und Spannungsversorgungen zu verbauen.

Wichtig ist, dass alle eingebauten Komponenten einen erweiterten Temperaturbereich besitzen und eine hohe Luftfeuchtigkeit keine Probleme verursacht.

1.1.2.2 Schranke

Bei diesem Projekt sind 2 Schranken im Einfahrtsbereich notwendig die in redundanter Ausführung hintereinander montiert sind. Eine der beiden Schranken dient dabei immer nur als Rückfallebene, d.h. falls eine Schranke eine Störung oder eine manuelle Sperre meldet, wird die übergeordnete Steuerung nur die aktive Schranke steuern.

Für die übergeordnete Steuerung werden verschiedene Schnittstellen zur Verfügung gestellt um die Schranken zu öffnen oder auch zu schließen.

Des Weiteren muss es einem Techniker möglich sein diese Schranke per Schlüsselschalter vor Ort direkt zu steuern. Diese Schlüsselschalter-Funktion dient hauptsächlich Wartungszwecken oder Notfällen mit Einsatzfahrzeugen.

Folgende Funktionen sind bei aktuellen Schranken verfügbar:

- Ansteuerung der Schranke von einer übergeordneten Steuerung
- Auslesen des Status der Schranke

- Geöffnet/Geschlossen
- Störung der Schranke
- Schlüsselschalter aktiv
- Induktionsschleife- und Lasersensor-Status damit die Schranke nicht bei durchfahrenden Fahrzeugen schließt

Die Ansteuerung der Schranke sollte über potentialfreie Kontakte oder über ein offenes industrielles Kommunikationsprotokoll über TCP/IP [2] möglich sein.

1.1.3 Parkstandsreihe

Jede Parkstandreihe verfügt bei der Einfahrt über einen Überkopfträger mit einer intelligenten LED-Anzeigen mit zwei Leuchtbildern. Die jeweiligen LED-Anzeigen müssen für dieses Projekt nur zwei verschieden Leuchtbilder anzeigen und dem Fahrer die zugewiesene Parkstandsreihe signalisieren, die nicht zugewiesenen signalisieren eine Sperre

- Parkstandsreihe für zugewiesenes Fahrzeug offen: „Grüner Pfeil“ LED-Leuchtbild
- Parkstandsreihe gesperrt: „Rotes X“ LED-Leuchtbild

Diese LED-Zeichen können von übergeordneten Steuerungen, je nach Hersteller, mit verschiedenen Varianten angesteuert werden:

- Proprietäres Protokoll über RS485[3] oder RS232[4]
- industrielles Kommunikationsprotokoll über RS485 oder 232
- Proprietäres Protokoll über TCP/IP
- industrielles Kommunikationsprotokoll über TCP/IP
- Potentialfreie Kontakte(24V DC)

Generell müssen diese Zeichen auch über mindestens einen Fehlerkontakt verfügen, damit zum Beispiel der Betreiber des Parkplatzes weiß, ob eine LED-Anzeige eine Störung aufweist.

[2] TCP/IP ist eine Familie von Netzwerkprotokollen (siehe https://www.w3.org/People/Frystyk/thesis/TcpIp.htmll)

[3] RS485 ist ein Industriestandard für asynchrone serielle Übertragung (siehe https://www.lammertbies.nl/comm/info/RS-485.html)

[4] RS232 ist ein Industriestandard für serielle Übertragung (siehe https://www.lammertbies.nl/comm/info/RS-232_specs.html)

1.2 Steuerungs- und Visualisierungssystem

Moderne Steuerungs- und Visualisierungssysteme unterstützen einen skalierbaren Aufbau und es können alle notwendigen Funktionen von einem zentralen Server ausgeführt werden.

Die Systemüberwachung ist mit einem zentralen Serversystem zu realisieren, muss aber zugleich die Nutzung von verteilten Hardware-Ressourcen und Anlageteilen unterstützen.

Folgende Funktionen sollten mindestens zur Verfügung stehen und sind bei modernen Systemen Standardfunktionen:

- Redundante Architektur
- Integriertes SCADA[5] System
- Integriertes Soft SPS[6] System
- Direkte Anbindung an redundanter Datenbank
- Webservice Anbindung
- Standardisierte Kommunikationsprotokolle
- Zentrale Steuerungseinheit
- Einsatz unterschiedlicher Hardwareplattformen
- Client-Server Architektur
- Soft SPS basierend auf IEC 61131-3[7]

1.3 Webservice

Um mit verschiedenen Systemen unterschiedlicher Hersteller in einem geschlossenen System über TCP/IP zu kommunizieren sind Webservices ein einfacher Weg eine standardisierte Schnittstelle zur Verfügung zu stellen. Für Webservices gibt es schon eine Menge vorgefertigter Bibliotheken in den verschiedensten Programmiersprachen. Für Systeme oder Firmen, die nicht im industriellen Umfeld tätig sind, bieten Webservices einen zusätzlichen Vorteil, da sie einen einfachen Weg darstellen um zwischen verschiedenen Systemen Daten auszutauschen.

Grundsätzlich haben Webservices einige Nachteile, weil sie oft nicht für schnelle Anwendungen geeignet sind und auch als sehr unsicher gelten. Für dieses Projekt spielen diese Nachteile aber keine Rolle da folgende Gegebenheiten vorherrschen.

---

[5] SCADA ist ein Softwaresystem für Datensammeln, Visualisieren und Steuern der angeschlossenen Anlagenteile

[6] Eine Soft SPS ist eine speicherprogrammierbare Steuerung in Software

[7] IEC 61131-3 ist ein offener Internationaler Standard für SPS Programmierung

- Keine zeitkritischen Funktionen nötig
- Geschlossenes Netzwerk ohne Anbindung nach außen

Entscheidend ist, sich auf eine Möglichkeit der Implementierung zu einigen, damit am Anfang des Projekts die zu übertragenden Daten in der definierten Form aufbereitet werden können. Im nachfolgenden Unterkapitel wird, die derzeit gängige REST Implementierung näher beschrieben und die Möglichkeiten in welcher Form die Daten übertragen werden müssen.

1.3.1 REST

REST ist die Abkürzung für Representational State Transfer und nach der Homepage für die REST API lautet die Definition aus der Quelle [RESTAPI2018] wie folgt:

„Der als REST bezeichnete Architekturansatz beschreibt, wie verteilte Systeme miteinander kommunizieren können. In diesem Sinne stellt eine REST API[8] eine Alternative zu anderen Schnittstellen wie SOAP[9] oder WSDL[10] dar. REST selbst ist dabei allerdings weder Protokoll noch Standard. Als „RESTful“ charakterisierte Implementierungen der Architektur bedienen sich allerdings standardisierter Verfahren, wie HTTP/S[11], URI[12], JSON oder XML[13].“

[8] API ist eine Programmierschnittstelle

[9] SOAP ist ein Netzwerkprotokoll (siehe https://www.w3.org/TR/soap/)

[10] Eine WSDL beschreibt einen Netzwerkservice (siehe https://www.w3.org/TR/wsdl/)

[11] HTTP/S ist ein Protokoll zur Übertragung von Daten (siehe http://www.w3.org/Protocols/)

[12] Ein URI dient zur Identifizierung von abstrakten oder physischen Ressourcen (siehe http://www.w3.org/Addressing/)

[13] XML ist eine Auszeichnungssprache für hierarchische Darstellung (siehe https://www.w3.org/XML/)

Folgende Methoden werden von REST zur Verfügung gestellt, um die Maschine zu Maschinen Kommunikation zu vereinfachen:

HTTP-Methoden	Beschreibung
GET	Fordert die angegebene Ressource vom Server an. GET weist keine Nebeneffekte auf. Der Zustand am Server wird nicht verändert, weshalb GET als sicher bezeichnet wird.
DELETE	Löscht die angegebene Ressource.
POST	Fügt eine neue (Sub-)Ressource unterhalb der angegebenen Ressource ein. Da die neue Ressource noch keinen URI besitzt, adressiert der URI die übergeordnete Ressource. Als Ergebnis wird der neue Ressourcenlink dem Client zurückgegeben. POST kann im weiteren Sinne auch dazu verwendet werden, Operationen abzubilden, die von keiner anderen Methode abgedeckt werden.
PUT	Die angegebene Ressource wird angelegt. Wenn die Ressource bereits existiert, wird sie geändert.

Abbildung 3: REST Methoden aus Quelle [Methoden2018]

1.3.1.1 Datenformate

Die Daten die über die REST Anfragen gesendet oder beantwortet werden müssen, können je nach benutzter Bibliothek entweder XML oder JSON[14] Format besitzen. Bei aktuelleren Implementierungsvarianten wird fast ausschließlich nur mehr das JSON Format benutzt.

Folgende Vorteile bringt das JSON Format mit sich:

- Verschiedene Datentypen
  - Bool
  - Integer
  - String
  - Float
  - Date-Time
- Verschachtelung
- Arrays
- Direktes Importieren der Daten in Datenbanken
- Direktes Exportieren der Daten aus Datenbanken

14 JSON ist eine kompaktes Datenformat für Datenaustausch (siehe https://json.org)

2 Generelle Rahmenanforderungen

Im vorangegangenen Kapitel wurde Grundlegendes des zu integrierenden intelligenten LKW-Kolonnenparksystems und Webservices sowie deren Notwendigkeit und Vorteile diskutiert. In diesem Kapitel erfolgen die Darstellung des Istzustandes und die Präzisierung der Aufgabenstellung.

2.1 Steuerungs- und Visualisierungssystem

Das Steuerung und Visualisierungssystem, muss in redundanter Ausführung auf virtuellen Windows Server 2016 [15] Maschinen installiert werden. Im Bereich der Hardware laufen die physikalischen Server mit dem Virtualisierungsbetriebssystem VMware vSphere esxi 6.5[16]. Auf den physikalischen Servern werden auch noch die virtuellen Maschinen des Herstellers des Kolonnenparken-Kontrollers ausgeführt, um die Anzahl der verwendeten Hardware zu minimieren.

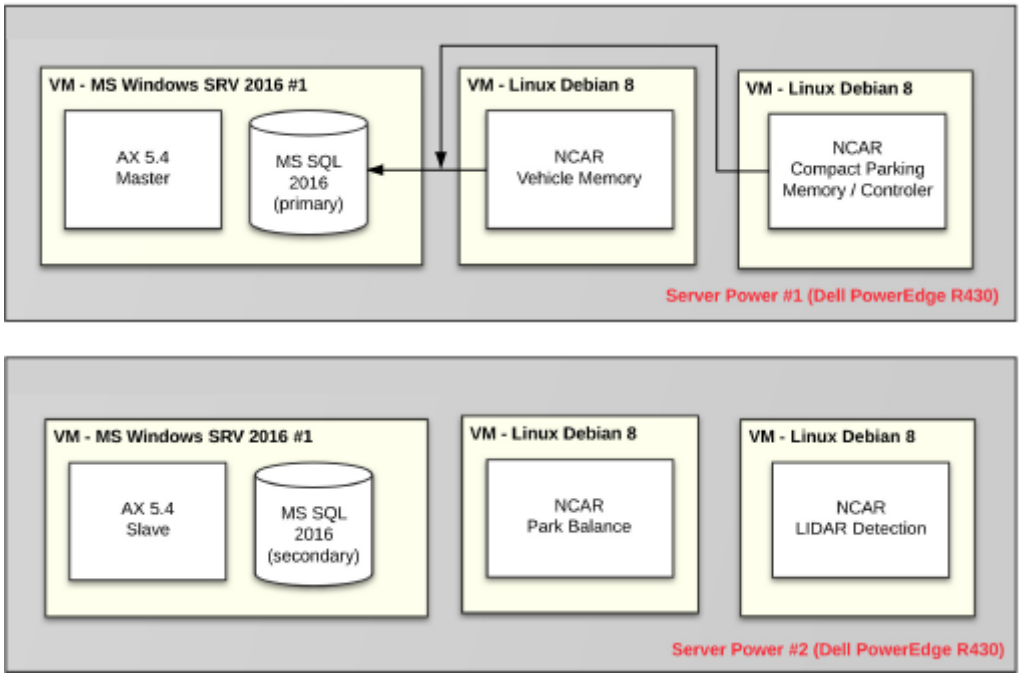


Abbildung 4: Systemaufbau Server

[15] Ist ein gängiges Betriebssystem für Server (siehe https://de.wikipedia.org/wiki/Microsoft_Windows_Server_2012)

[16] Ist ein Betriebssystem explizit für Virtualisierung Lösungen (siehe https://docs.vmware.com/de/VMware-vSphere/index.html)

Das zu installierente Steuerungs- und Visualisierungssystem kommt aus dem Hause AutomationX[17] und ist .NET[18] basierend. Aufgrund der offenen Standards und der Flexibilität des verwendeten Systems können in Bezug auf die Integration des intelligenten LKW-Kolonnenparken die vielen Vorteile genutzt werden. Es stehen alle Werkzeuge zur Verfügung um diese Integration und Neu-Entwicklungen durchzuführen. Dadurch spart sich der Kunde eine Menge Lizenz-, Aufwands- und Entwicklungskosten. Als Datenbank wird eine MS-SQL Server 2016[19] Instanz installiert. Das eingesetzte AutomationX-System kann eine MS-SQL Datenbank direkt integrieren.

2.1.1 Allgemeine Funktion des Steuerungs- und Visualisierungssystems

Das Steuerungs- und Visualisierungssystem muss mit allen Anlagenteilen des intelligenten LKW-Kolonnenparkens kommunizieren und eine Server-Client Architektur zur Verfügung stellen.

Folgende Komponenten müssen gesteuert oder verwaltet werden:

- MS SQL 2016 Datenbank mit „Always On Avaibility Cluster[20]“ Redundanz
- Visualisierung am Bedienplatz
- Visualisierung am Touch-Monitor am Einfahrtsbereich
- Steuerung Schranke
- Steuerung LED-Anzeigen über Parkstandsreihen
- Integrierte REST Schnittstelle für folgende Methoden
    - GET
    - POST
    - PUT
    - DELETE
- Verarbeiten von JSON Daten

Es gibt eine direkte Verbindung zu einer Microsoft SQL Datenbank, das ist eine gute Voraussetzung für das Berichtswesen und das Abbilden des Parkplatzes, denn jeder einzelne Bedienereingriff, Alarm oder Zustandsänderung wird sofort in der Datenbank gespeichert und kann durch Datenbankabfragen zur jeder beliebigen Zeit in Form eines Berichts zur Verfügung gestellt werden.

---

[17] AutomationX ist ein international tätiges Technologieunternehmen mit einer eigenen Hard und Software-pa-lette (siehe www.automationx.com)

[18] .NET ist eine Softwareplattform (siehe https://www.microsoft.com/net/)

[19] MS-SQL Server 2016 ist eine relationale Datenbank (siehe https://www.microsoft.com/de-de/sql-server/sql-server-2016)

[20] Always On Avaibility Cluster ist eine Form der Redundanz, die eine MS-SQL Datenbank zur Verfügung stellt.

Die Aufgabe des Steuerungs- und Visualisierungssystems in diesem Projekt ist die vom Kolonnenparken-Kontroller zur Verfügung gestellten Schnittstellen zu integrieren und dann mit den zur Verfügung gestellten Werkzeugen und Daten (z.B. aus dem Kolonnenparken-Kontroller) eine zentrale Bedienoberfläche für den Bediener und Fahrer zu erzeugen. Für den Bediener ist es nicht mehr nötig verschiedene Applikationen zu benutzen, er kann das komplette Parkplatzsystem vom Steuerungs- und Visualisierungssystem aus verwalten.

2.1.2 Verfügbarkeit des Steuerungs- und Visualisierungssystems

Das Serversystem muss für einen 24/7-Betrieb ausgelegt sein.

Um eine hohe Verfügbarkeit des Systems zu sichern, sind die Server in Hot-Standby Redundanz auszuführen (Master-Slave-Betrieb). Dabei hat der Master Zugriff auf die Prozessdaten, während der Slave ohne Nutzung seines Prozessanschlusses sich zum aktuellen Datenbestand des Masters synchronisiert. Prozessausgaben erfolgen nur über den jeweiligen Master.

Bei Ausfall des Masters erfolgt ein stoßfreier Wechsel und der bis dahin als Slave agierende Server übernimmt die Master-Funktion. Somit ist der Zugriff auf Daten bzw. Funktionen jederzeit gewährleistet.

2.2 Feldbussysteme

Bei diesem Projekt kommen nicht nur Anlagenteile mit einer Webservice-Schnittstelle zum Einsatz. Im Normalfall werden die Anlagenteile in der Feldebene über Ein- bzw. Ausgabebaugruppen der SPS oder über Feldbusschnittstellen erfasst und in der SPS für das Leitsystem aufbereitet.

Das neue Steuerungs- und Visualisierungssystem muss in der Lage sein Sensoren mit Feldbusschnittstellen direkt unter Umgehung der Steuerungen in das Leitsystem zu integrieren. Die Rahmenanforderung ist, dass dieses mit offenen industriellen Kommunikationsprotokollen für Feldbussysteme zu erfolgen hat. Anhand der Anforderungen des AG und der ausgelieferten Anlagenteile gibt es 3 Möglichkeiten diese zu steuern:

- Ansteuerung über potentialfreie Kontakte
- Ansteuerung über offenen Industriestandard über TCP/IP
    - Modbus TCP[21]
    - OPC/UA[22]
    - IEC 60870-5-104[23]

[21] Modbus TCP ist ein offenes industrielles Kommunikationsprotokoll über TCP/IP (siehe http://www.modbus.org/)

[22] OPC/UA ist ein industrielles Kommunikationsprotokoll (siehe https://opcfoundation.org/)

    - Profinet[24]
- Ansteuerung über offenen Industriestandard über serielle Schnittstellen
    - Modbus RTU[25]
    - Profibus[26]

Die zu übertragenden und entgegenzunehmenden Informationen und Daten sind in einer Datenpunktliste festgelegt. Dies ist die Grundlage für die Verknüpfung zwischen dem Steuerungs- und Visualisierungssystem und den Feldbuskomponenten. Wesentliche Inhalte umfassen die Schaltung der Schranke, der Anlage sowie die Übertragung von Zustandsdaten der LED-Anzeige.

Eine zusätzliche wichtige Anforderung ist auch die Kosten und das benötigte Equipment so gering wie möglich zu halten.

2.3 Der Bedien-Rechner

Es gibt in diesem Projekt mehrere Clients, die je nach Anforderung unterteilt sind. Ein Bedienplatz dient dem Betreiber des Parkplatzes, die Touch-Monitore dienen den Fahrern die den Parkplatz benutzen wollen, um diverse Eingaben zu machen.

Diesbezüglich müssen mehrere Prozessbilder und Objekte erstellt werden, damit diese gewünschten Funktionen den unterschiedlichen Bediener zu Verfügung stehen.

2.3.1 Allgemeine Funktion des Bedien-Rechners

Der Bedienrechner steht in einem kleinen Kabelhaus neben dem Parkplatz. Der Bedienplatz hat 2 24“ FullHD[27] Monitore, auf denen die Visualisierung den Bedienern zur Verfügung gestellt wird. Als Betriebssystem wird Windows 10 benutzt.

Vom Bedien-Rechner müssen mindestens folgende Funktionen für den Bediener zur Verfügung gestellt werden:

- Übersicht über den Belegungsstatus des Parkplatzes
- Visualisierung über 2 Monitore
- Anzeige des aktuellen Status der Anlagenteile
- Manueller Eingriff in die Steuerung

---

[23] IEC 60870-5-104 ist ein industrielles Kommunikationsprotokoll

[24] Profinet ist ein industrieller Feldbus (siehe https://www.profibus.com/technology/profinet/)

[25] Modbus RTU ist ein industrielles Kommunikationsprotokoll über seriellen Bus (siehe http://www.modbus.org/)

[26] Profibus ist ein industrieller Feldbus (siehe https://www.profibus.com/technology/profibus/)

[27] FullHD ist eine Bildschirmauflösung von 1920x1080 Pixel

- Schalten der LED-Anzeige
- Schalten der Schranke
- Detailinfo der Fahrzeuge
- Liste aktueller Alarme der Anlagenteile
- Ändern der Konfigurationsparameter
- Verschieden Benutzerlevel (Autorisierungsstufen)

2.3.2 Allgemeine Funktion des Einfahrtsterminals

Am Einfahrtsterminal sollen 2 19“ Touch-Monitore mit einer Auflösung von 1280x1024 für den Fahrer zur Verfügung stehen. Die Touch Monitore müssen folgende Funktionen erfüllen:

- 24/7 Betrieb
- Erhöhter Temperatur Bereich von -30 C°-+80 C°
- Wasserfest
- Kapazitiver Touch-Monitor
- Hohe Helligkeit
- Hoher Blickwinkel

Die Touch-Monitore haben je einen abgesetzten Industrie PC im Innenraum des Einfahrtsterminals mit HDMI und USB angeschlossen. Der Industrie PC muss folgende Rahmenanforderungen erfüllen:

- Windows 10 IOT[28]
- 24/7 Betrieb
- Lüfterlos
- Visualisierung des Steuerungs- und Visualisierungssystems anzeigen
- Eingaben verwalten (Abfahrtszeit)
- USB-Ticket-Drucker steuern
- Steuern der Schranke

Die Visualisierung für die Navigation muss mehrsprachig sein und wird vom Kunden vorgegeben und auch patentiert.

[28] Windows 10 IOT ist ein Betriebssystem für „Embedded Systems“ (siehe https://docs.microsoft.com/en-us/windows/iot-core/windows-iot)

3 Systemkonzept

In diesem Kapitel wird das Systemkonzept für die Schnittstellen, sowie die hierarchische Struktur des Datenaustauschs bis hin zur Visualisierung dargestellt. Es beinhaltet einen Überblick über den Aufbau des JSON Formats, Abläufe der einzelnen Methoden und eine Erläuterung von relevanten Anlageteilen. Aufgaben des intelligenten LKW-Kolonnenparksystems mit dem Steuerungs- und Visualisierungssystem werden aufgezeigt und das Zusammenspiel dieser Systeme genauer betrachtet

3.1 Aufbau JSON

Durch die Neuheit des eingesetzten Systems gibt es noch keine definierte Schnittstelle und auch keine Erfahrungswerte von anderen Parkplätzen, welche Daten ausgetauscht werden müssen. Deshalb wird zuerst Format, Feldnamen und Datentypen mit dem Hersteller des Kolonnenparken-Kontrollers definiert. Die verfügbaren Daten wurden in 4 Gruppen aufgeteilt um bei spezifischen Anfragen die gewünschten Daten detaillierter zu empfangen. Bezüglich der Feldnamen wurde vereinbart, dass die „lowerCamelCase“-[29] Notation für die Namensgebung benutzt wird, weil das AutomationX-System keine Sonderzeichen in einem JSON Datenformat erlaubt und die Übersichtlichkeit erhalten bleibt.

3.1.1 Parkplatzspeicher

Für die Visualisierung des Parkplatzes und für die Steuerung der LED-Anzeige über den Parkstandsreihen müssen mindestens folgende Informationen zur Verfügung stehen, damit eine genaue Visualisierung oder Steuerung der einzelnen LED-Anzeigen erfolgen kann. Den einzelnen Parkstandsreihen sind auch die Fahrzeug-IDs angehängt, die auf den abgefragten Parkstandsreihen parken oder fahren.

Eigenschaft	**Datentyp**	**Beschreibung**
queue	Array	
id	String	Eindeutige ID
length	float	Länge der Parkstandsreihe
vehicles	vehicle [Array]	Tabelle vehicle
freeDistance	float	Noch freie Parkplatzlänge

[29] LowerCamelCase ist eine zusammengesetzte Namensgebung (siehe http://wiki.c2.com/?LowerCamelCase)

isBooked	Boolean	Reihe ist gebucht
isClosed	Boolean	Reihe ist geschlossen
isLocked	Boolean	

Tabelle 1: JSON Aufbau Parkplatzspeicher

Eigenschaft	Datentyp	Beschreibung
vehicle	Array	
id	String	Eindeutige ID
position	float	Position des Fahrzeugs
timeDeparture	DateTime	Eingebene Abfahrtszeit
overtime	UInt32	Überzeit
parkingTime	DateTime	Startzeit des Parkens
isWildparker	Boolean	Falschparker

Tabelle 2: JSON Aufbau Zusatz Fahrzeuginformation

Weiterführende Informationen zum JSON Aufbau gibt es unter [DokuKolopa2018].

3.1.2 Fahrzeugspeicher

Der Fahrzeugspeicher beinhaltet die Stammdaten der einzelnen Fahrzeuge. In der ersten Stufe der Visualisierung im Übersichtsbild werden nur Position, Länge und Art des Fahrzeuges visualisiert. Bei einem Links-Klick auf das gewünschte Fahrzeug werden dann die näheren Infos geladen und dem Bediener dargestellt.

Eigenschaft	Datentyp	Beschreibung
vehicles	Objekt	
class	UInt32	Fahrzeugklasse(PKW, LKW …)
color	string	Farbe des Fahrzeugs
country	string	Land auf dem Kennzeichen
hash[30]	string	Hashfunktion

[30] Der Hashwert dient als Prüfsumme

hazardIdentificationNumber	string	Gefahrengutnummer
height	float	Höhe des Fahrzeugs
id	string	Eindeutige ID
length	float	Länge des gesamten Fahrzeugs
maker	string	Hersteller des Fahrzeugs
model	string	Model des Fahrzeugs
number	string	Kennzeichen
silhouette	string	Silhouette des Fahrzeugs
width	float	Breite des Fahrzeugs

Tabelle 3: JSON Aufbau Fahrzeuginformation

Weiterführende Informationen zum JSON Aufbau gibt es unter [DokuKolopa2018].

3.1.3 Konfigurations-Parameter

Damit der Bediener die Möglichkeit besitzt Konfigurations-Parameter für den Kolonnen-parken-Kontroller anzupassen, werden einige Konfigurations-Parameter zur Verfügung gestellt. Jeder übergeordnete Parameter besitzt ein eigenes JSON Objekt. In der unten angeführten Tabelle werden z.B. die Parameter für die Zwölf-Stunden-Regel näher erläutert.

Eigenschaft	**Datentyp**	**Beschreibung**
twelveHoursRule	Objekt	
checkInterval	Int32	Intervall in Sekunden für die Prüfung
prolongPeriod	Int32	Verlängern des Zeitraums
overTimesToCallOperator	Int32	Überzeit um eine Alarmierung durchzuführen
overTimesToCloseQueue	Int32	Überzeit für die Sperrung des Parkstands-reihe
timeBuffer	Int32	Zeitpuffer

Tabelle 4: JSON Aufbau Zwölf Stunden Regel

Weiterführende Informationen zum JSON Aufbau gibt es unter [DokuKolopa2018].

3.2 REST Methoden und Endpunkte

Die REST API besitzt eine Server-Client Architektur. Die Client-Funktion der REST API wird auf dem Steuerungs- und Visualisierungssystem ausgeführt. Die Server-Funktion wird am Kolonnenparken-Kontroller ausgeführt um mit definierten Methoden zyklische Abfragen zu empfangen, oder gegebenenfalls bei Auftreten eines Ereignisses eine Änderung zu senden. Es wurde vereinbart, dass für jedes JSON Format ein eigener Endpunkt für die jeweiligen REST Methoden erstellt wird. Nicht jede Information hat die gleiche Priorität oder muss immer verfügbar sein, deshalb kann in verschiedenen Zyklen abgefragt werden um die Systemlast zu minimieren.

Eine Authentifizierung ist nicht notwendig da hier in einem geschlossen Netzwerk kommuniziert wird.

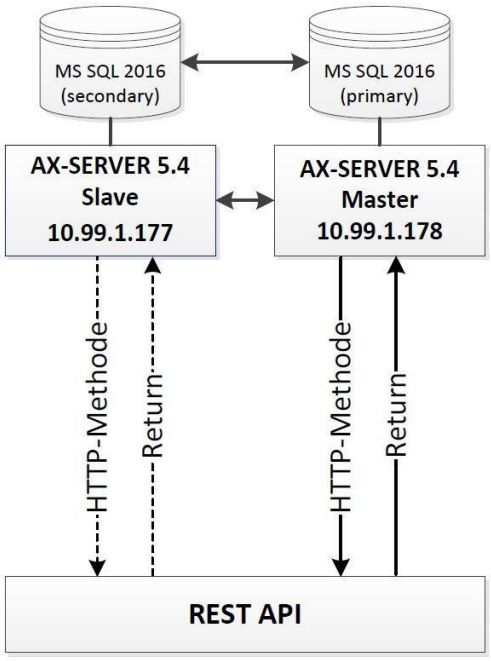


Abbildung 5: Konzept REST API

In den folgenden Unterkapiteln werden die verschiedenen REST Endpunkte mit den jeweiligen Methoden beschrieben.

3.2.1 Allgemeine Parkplatzinformation

Für die Abfrage zur Übermittlung der Daten für die Visualisierung des gesamten Parkplatzes wird ein eigener Endpunkt zur Verfügung gestellt. Diese allgemeinen Parkplatzinformationen können nur in lesender Verbindung abgefragt werden, um den aktuellen Zustand des gesamten Parkplatzes zu erfassen. Der Zyklus der Abfrage wird im 5 Sekunden Takt abgefragt, d.h. dass die Position des Fahrzeugs auf dem Parkplatz der Visualisierung alle 5 Sekunden aktualisiert wird. Dadurch kann eine Fahrzeugverfolgung ohne Live Kamerabild für den Bediener auf einem Prozessbild durchgeführt werden.

Request	
HTTP-Methode	GET
Host/Url	http://10.99.1.200:4000/parking
URL-Parameter	entfällt
HTTP-Body	entfällt

Tabelle 5: REST GET Methode Parkplatzinformation

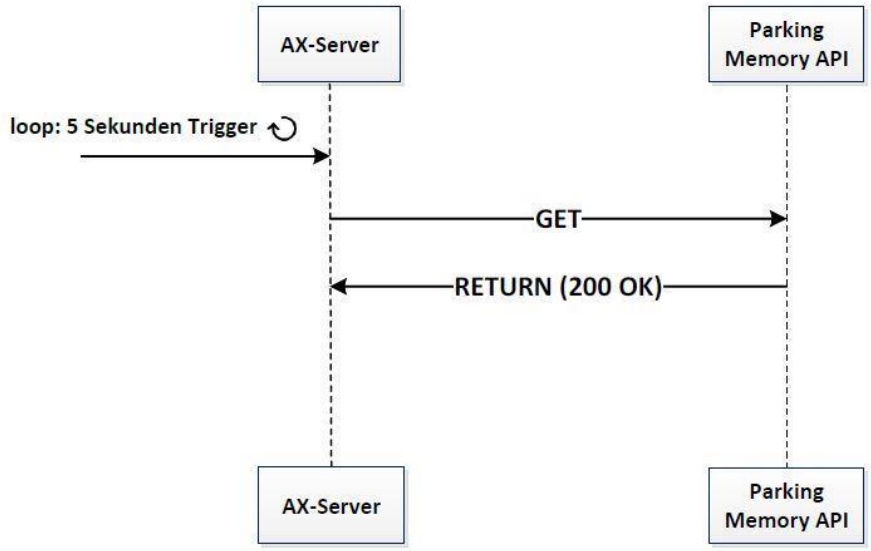


Abbildung 6: REST GET Parkplatzinformation

3.2.2 Allgemeine Fahrzeuginformation

Um Detailinformationen für die parkenden oder sich bewegenden Fahrzeuge, die sich auf dem Parkplatz aufhalten, abzufragen, wurde ein eigener Endpunkt realisiert. Diese Daten werden in einem Zyklus von 7 Sekunden für die Visualisierung aktualisiert. Die Datensätze werden falls der Parkplatz ausgebaut, oder das System auf einem größeren Parkplatz eingesetzt wird mit einem eigenen Parameter versehen, um ein Limit von Datensätzen zu erhalten.

Request			
HTTP-Methode	GET		
Host/Url	http://10.99.1.200:8066/vehicles?limit=999		
URL-Parameter	**Parameter**	**Datentyp**	**Beschreibung**

	limit	number	Maximale Anzahl an Datensätzen
HTTP-Body	entfällt		

Tabelle 6: REST GET Methode Fahrzeuginformation

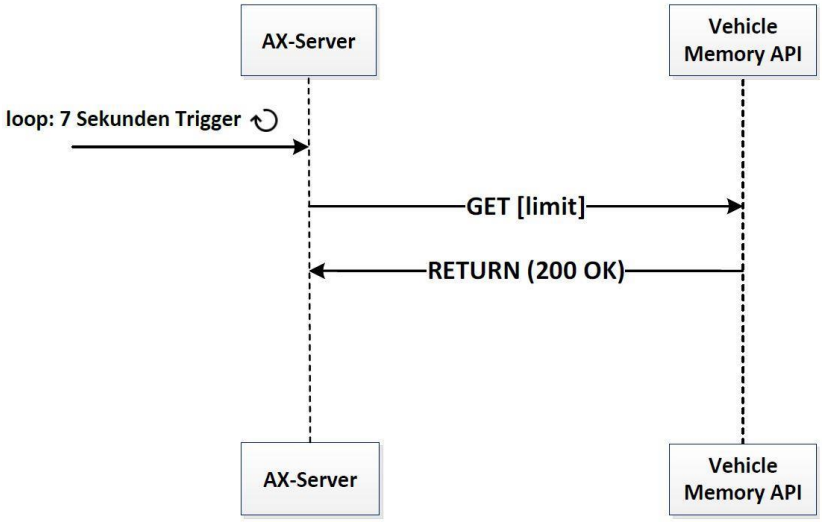


Abbildung 7: REST GET Fahrzeuginformation

3.2.3 Allgemeine Konfigurations-Parameter

Der Kolonnenparken-Kontroller hat einige Konfigurations-Parameter um z.B. Fahrzeugerkennung, Verfolgung oder Abtastraten zu verändern. Für die Konfigurations-Parameter gibt es übergeordnete Konfigurations-Objekte und in diesen Objekten sind dann diverse Parameter verschachtelt.

Request	
HTTP-Methode	GET
Host/Url	http://10.99.1.200:5300/config
URL-Parameter	entfällt
HTTP-Body	entfällt

Tabelle 7: REST GET Methode Konfigurations-Parameter

Der Unterschied zu den anderen Endpunkten ist, dass hier nicht nur derzeitige Konfigurationsparameter abgefragt werden, sondern auch Werte verändert werden können. Eine Besonderheit stellt deshalb das Befüllen des HTTP-Bodies dar. Web-APIs funktionieren so, dass Daten entweder als URL-Parameter übergeben werden können oder über den Body. Das Speichern von Daten wird über eine Web API im Normalfall durch die HTTP-Methoden POST und PUT durchgeführt. In diesen Fällen sind Daten per Body zu übergeben. Das Setzen des Bodies mit Daten bei einem GET-Request ist grundsätzlich möglich

und wird von einigen Web APIs angeboten, ist aber nicht üblich und wird in der Spezifikation auch untersagt.

Request	
HTTP-Methode	POST
Host/Url	http://10.99.1.200:5300/config
URL-Parameter	entfällt

HTTP-Body	{
	"data": {
	"id": "Configuration-20180206-115832-727",
	"type": "Configuration",
	"version": "3.1",
	"time": "2018-02-06T11:58:32.727Z",
	"compactParking": {
	"controller": {
	"bookings": {
	"checkInterval": "30",
	"gracePeriod": "2700",
	"permissibleTimeDifference": "900",
	"timeBuffer": "600",
	"parkingGap": "1",
	"expirationTime": "900"
	}
	}
	}
	}
	}

Tabelle 8: REST POST Methode Konfigurations-Parameter

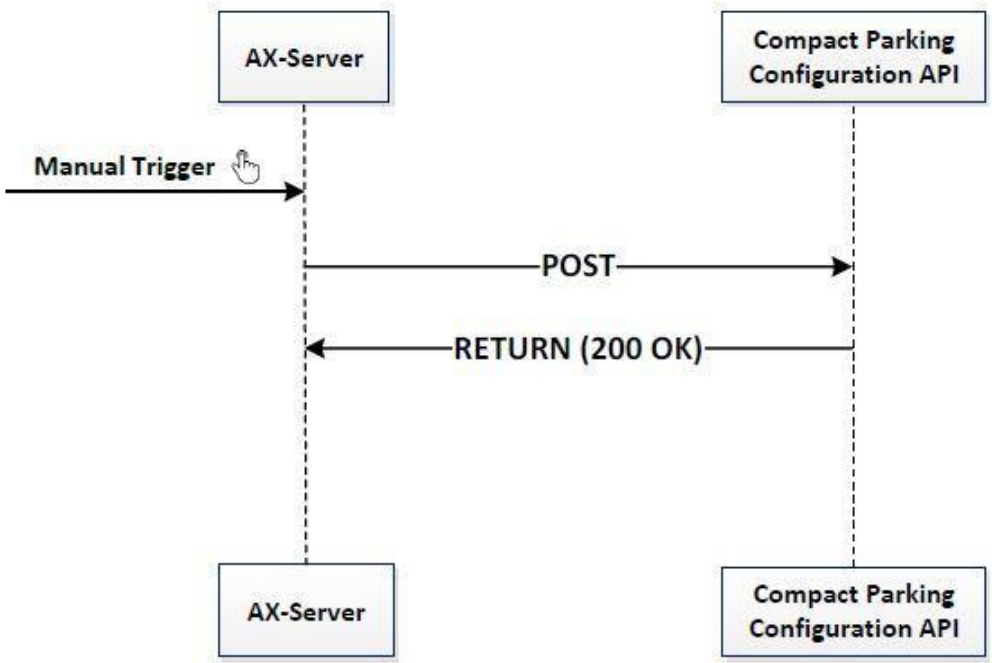


Abbildung 8: REST POST Methode Konfiguration

3.3 Aufbau Datenbank

Da das AutomationX-System die JSON Daten, die über die verschiedenen REST Methoden übermittelt werden, nicht direkt verarbeiten kann, müssen die Daten zuerst über das AutomationX-System in die Datenbank gespeichert werden, um dann der Visualisierung und Steuerung zur Verfügung zu stehen.

Die Verarbeitungszeit und der Aufwand erhöhen sich dadurch, aber da hier keine zeitkritische Anwendung gefordert ist, kann mit dieser Systematik trotzdem die gewünschte Lösung umgesetzt werden.

Die Struktur, Inhalt und Datentypen werden dem JSON Format der einzelnen Endpunkte nachempfunden, dadurch wird Erstellen, Auslesen und Verarbeiten der Daten schneller und verständlicher. Die zu abgelegten Daten weisen in den Datenbanktabellen eine Normalisierung der 5. Form auf.

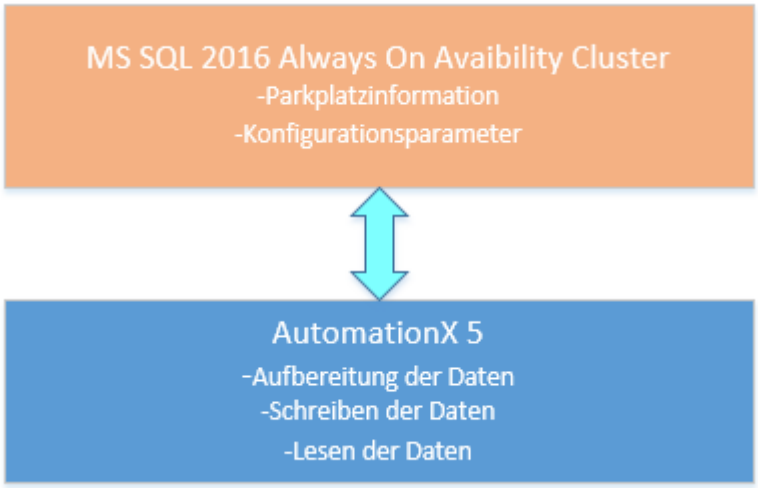


Abbildung 9: Datenbank-AX 5 Austausch

3.3.1 Aufbau Tabelle Parkplatzinformation

Für das Abbilden des Parkplatzes in der Visualisierung werden zwei Tabellen mit den Stammdaten und eine mit den Bewegungsdaten angelegt. Die jeweiligen Primärschlüssel der Stammdaten-Tabelle werden dann als zusammengesetzter Fremdschlüssel in der Bewegungsdatentabelle eingefügt, damit eine Verknüpfung zwischen Fahrzeug und Parkstandsreihe hergestellt werden kann. In der nachfolgenden Abbildung sind die Tabellennamen, Spaltennamen, Datentypen und Verknüpfungen dargestellt.

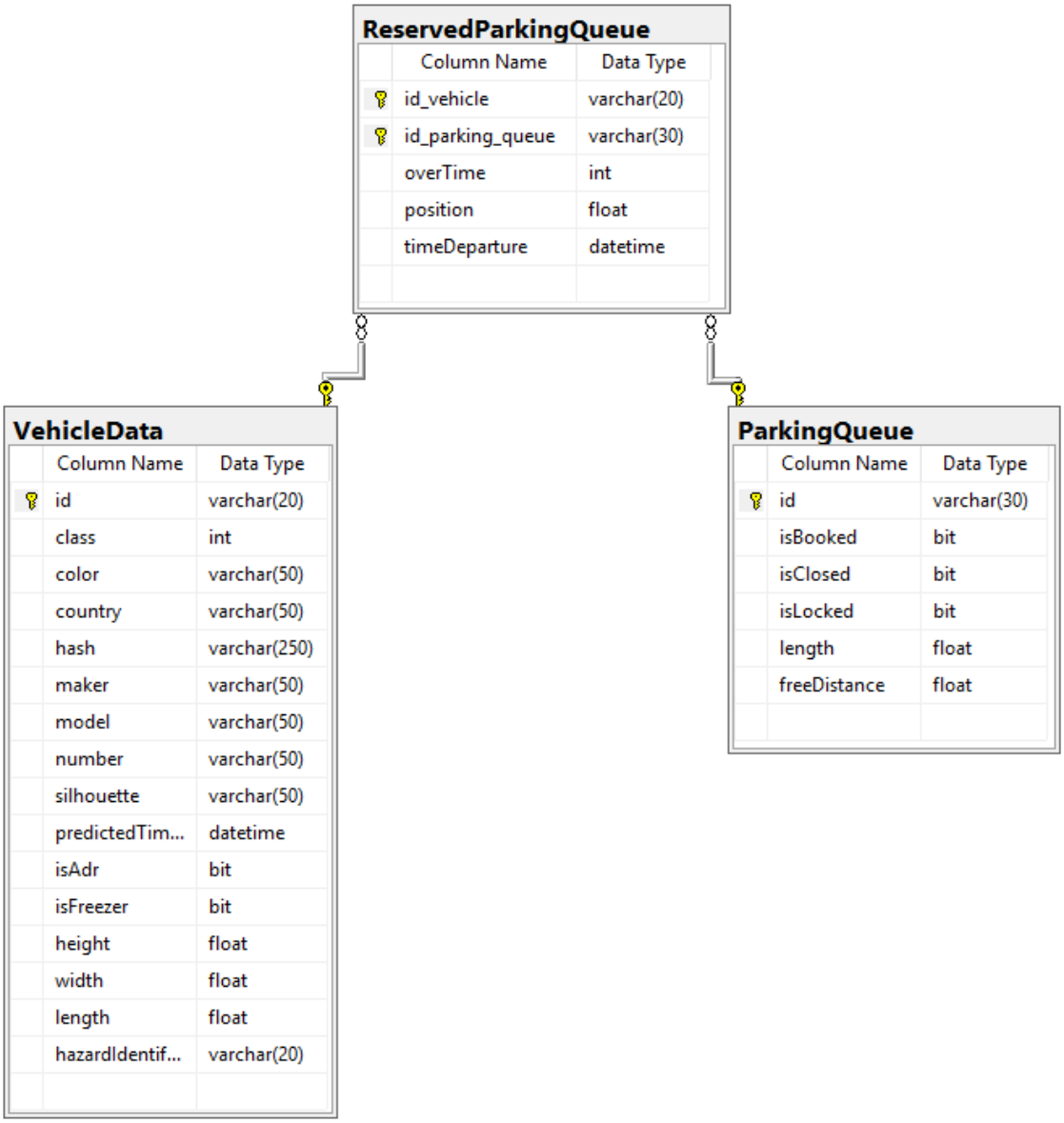


Abbildung 10: Datenbank Tabellen Parkplatzinformation

3.3.2 Aufbau Tabellen Konfiguration

Für die Konfigurations-Parameter wurde für jedes konfigurierbare Modul eine eigene Tabelle mit den jeweiligen Parametern angelegt. Sie können nicht in eine Tabelle zusammengefasst werden, weil hier keine Abhängigkeiten zwischen den unterschiedlichen Konfigurations-Parametern vorherrschen. In der nachfolgenden Abbildung sind die Tabellennamen, Spaltennamen, Datentypen und Verknüpfungen dargestellt.

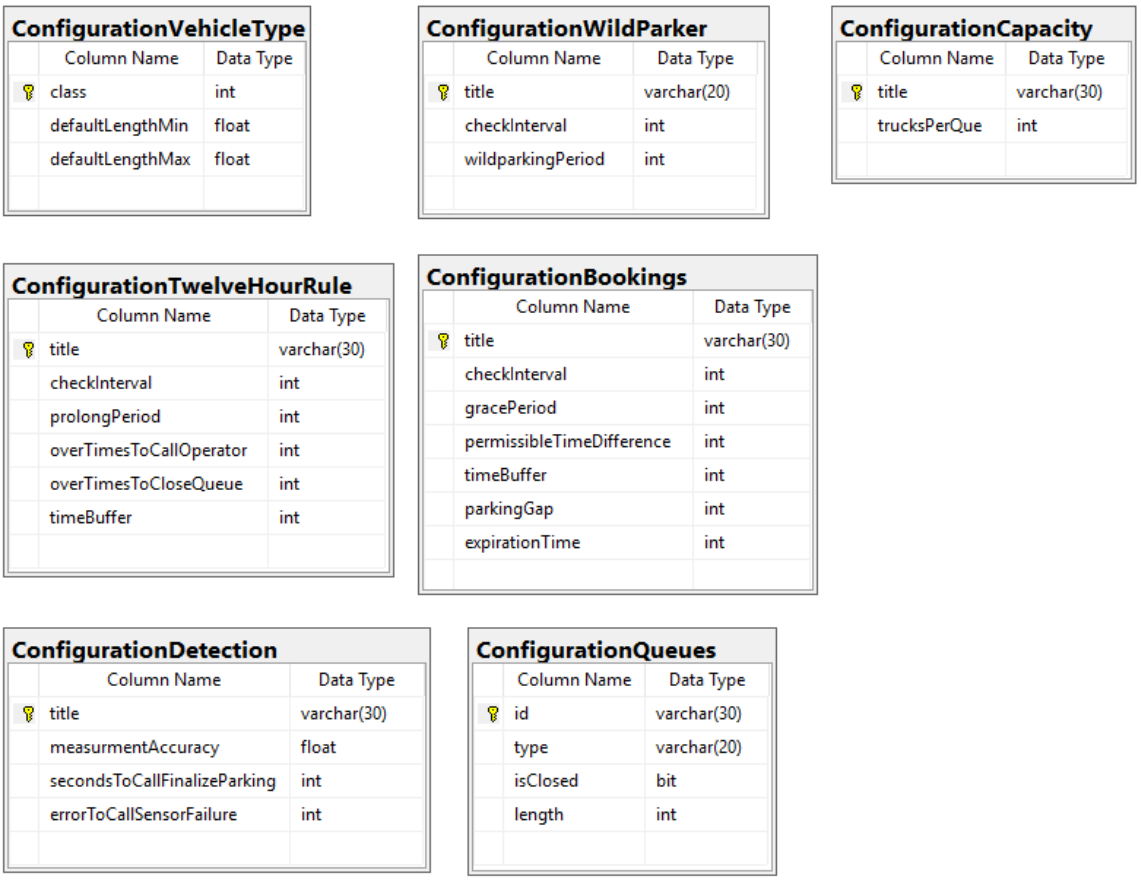


Abbildung 11: Datenbank Tabellen Konfiguration

3.4 Steuerung

In diesem Unterkapitel wird das Konzept für das Zusammenspiel der einzelnen Anlagenteile des Kolonnenparken-Kontrollers beschrieben.

3.4.1 Auslesen der Datenbankdaten

In Anbetracht der Daten des intelligenten Kolonnenparken-Kontrollers, die in der MS SQL Datenbank, wie in Kapitel 3.3 beschrieben, gespeichert und zyklisch aktualisiert werden, muss es der Steuerung möglich sein, sich mit der Datenbank zu verbinden und anhand der gespeicherten Daten einzelne Anlagenteile zu steuern. Die Abfrage der Daten von den jeweiligen SQL Tabellen durch die Steuerung wird zyklisch durchgeführt. Die Abfragezyklen sind nach abgeschlossener Aktualisierung der Daten in der Datenbank. Die

Synchronisierung der Abfragezyklen, der REST Methoden und das Einlesen der Daten für die Steuerung funktioniert mit setzen eines Merkers nach Abschluss der diversen REST Methoden. Danach startet die Abfragen der aktualisierten Datenbankinformation.

Die Abfragen werden mit T SQL[31] realisiert, weil dies ein wesentlicher Bestandteil von Microsoft SQL Server 2016 ist. Ein Vorteil von T SQL ist das hier lokale Variablen in den Queries[32] verwendet werden können. Durch diese Variablen können Abfragen objektorientiert in Klassen des AutomationX-Systems gespeichert werden. Bei der Instanzierung der einzelnen Klassen können die Variablen dann mit den richtigen Attributen beschrieben werden.

Einer dieser Einsatzgebiete kann die Steuerung der einzelnen LED-Anzeigen der Parkstandsreihen sein.

Die folgende Abbildung zeigt den Einsatz von T SQL mit einer Variablendeklaration.

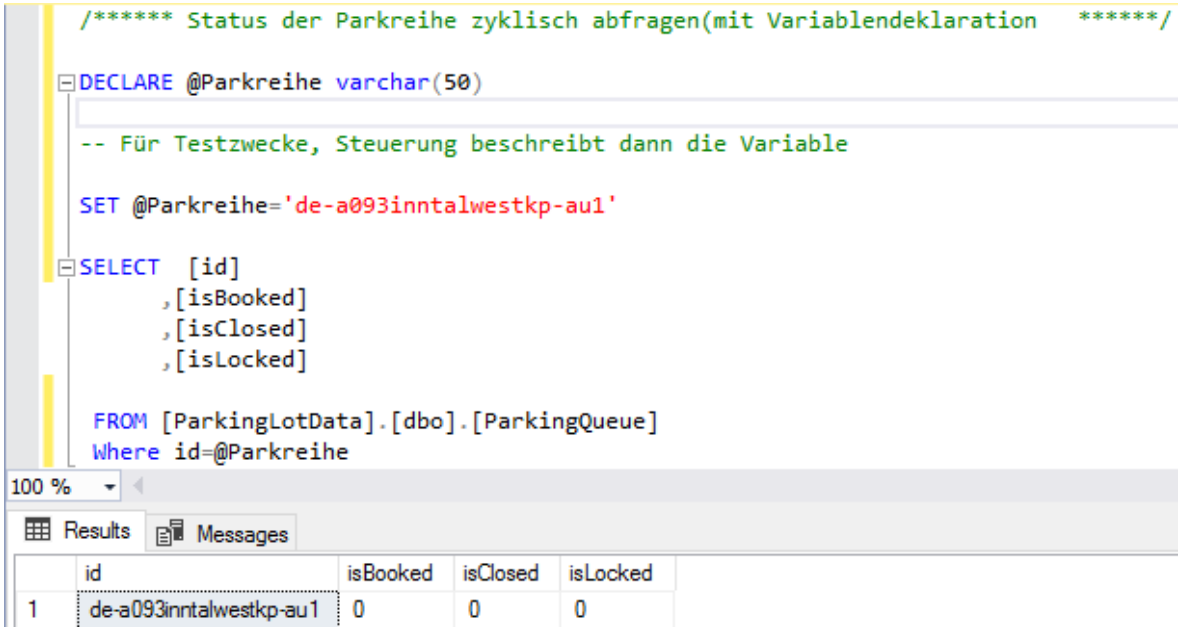


Abbildung 12: T SQL Abfrage mit Variable

3.4.2 Ansteuerung Schranke

Die Schranken im Einfahrtsbereich des Terminals, die vom Kunden geliefert wurden, sind mit zwei verschiedenen Technologien ausgestattet. Durch diese Technologien ist es möglich die Schranke mit folgenden Schnittstellen zu steuern:

- Modbus TCP

[31] T-SQL ist eine Erweiterung des SQL Standards (siehe https://searchsqlserver.techtarget.com/definition/T-SQL)

[32] Query ist eine Abfrage der Datenbank

- Potentialfreie Kontakte(DE/DA)

Über beide Schnittstellen stehen die gleichen Befehle und Meldungen zur Verfügung.

Die Steuerung muss anhand von Bediener Interaktionen oder Informationen vom Kolonnenparken-Kontroller die Schranke bei folgenden Ereignissen öffnen:

- Fahrer hat ein Ticket am Einfahrtsterminal erhalten
- Parkplatz ist voll
- Kolonnenparken-Kontroller hat einen Systemfehler
- Steuerungs- und Visualisierungssystem hat einen Systemfehler
- Einfahrtsterminal hat einen Systemfehler
- Manueller Eingriff eines Bedieners

Zusätzlich zum Öffnen/Schließen der Schranke wird auch ein Fehlermanagement umgesetzt, das bedeutet, dass die Schranke erst schließen darf, wenn die Lichtschranke der Schranke nicht unterbrochen ist.

3.4.2.1 Ansteuerung direkt über DE/DA oder Modbus TCP

Im Laufe der Evaluierung musste aufgrund der zwei verschiedenen Technologien für die Ansteuerung entschieden werden, wie die Schranke am Feldbussystem angeschlossen werden sollte. Das AutomationX-System kann Anlagenteile nicht direkt über DE/DA steuern und benötigt einen zusätzlichen Buskoppler mit DE/DA Anschlussklemmen als Hardware um die Schranke zu steuern. Die Lösung mit Buskopplern würde grundsätzlich die Rahmenanforderungen erfüllen, da das AutomationX-System über marktübliche Buskoppler, die DE/DA über Modbus TCP schalten können und zum gleichen Endergebnis kommen. Natürlich widerspricht dies aber der Ablehnung zusätzlicher Hardware und dadurch höherer Kosten und höherem Wartungs-Aufwand, deshalb macht es keinen Sinn die Schranke über DE/DA zu integrieren und dadurch zusätzlich den Buskoppler zu verwenden. Aufwand und Kosten würden sich nicht rentieren und es ist für zukünftige Erweiterungen nicht von Vorteil, weil die Schranke selbst einen Modbus TCP Anschluss besitzt und AutomationX dieses offene Industrieprotokoll auch unterstützt.

In der nachfolgenden Abbildung wird die schematische Ansteuerung der Schranke dargestellt.

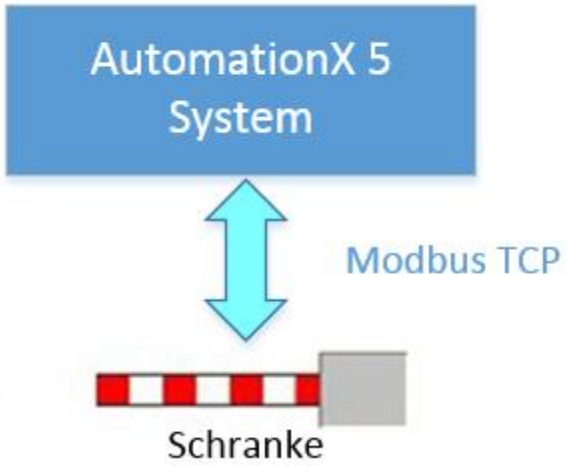


Abbildung 13: Konzept Ansteuerung Schranke

3.4.3 Ansteuerung LED-Anzeigen

Die eingesetzten LED-Anzeigen über den Parkstandsreihen unterstützen zwei Technologien zur Ansteuerung:

- Proprietäres Protokoll über RS485
- Potentialfreie Kontakte(DE/DA)

Mit diesen Ansteuerungsmöglichkeiten können folgende Meldungen und Befehle übermittelt werden:

- Meldung Parkstandsreihe gesperrt
- Meldung Parkstandsreihe offen
- Fehler LED-Anzeige
- Befehl Parkstandsreihe sperren
- Befehl Parkstandsreihe öffnen

Eine TCP/IP Ansteuerung ist nicht möglich, da diese Zeichen keinen RJ45[33] Anschluss besitzen.

Die Steuerung der LED-Anzeige muss direkt mit den zyklisch aktualisierten Daten des Kolonnenparken-Kontrollers in den jeweiligen Datenbanktabellen arbeiten. Der Fahrer bekommt über den Kolonnenparken-Kontroller eine Parkstandsreihe anhand der Abfahrtszeit zugewiesen. Diese Parkstandsreihe wird dem Fahrer anhand eines grünen Pfeils über der Einfahrt der Parkstandsreihe signalisiert. Der Grundzustand der LED-Anzeigen ist ein rotes X. Wenn der Tabelleneintrag „isBooked“ der Tabelle „Parking-Queue“(siehe Kapitel 3.3.1) mit dem boolschen Wert „True“ versehen ist, wird der Befehl „Parkstandsreihe öffnen“ an der LED-Anzeige angesteuert. Nachdem der Parkvorgang

[33] RJ 45 ist eine genormte Steckerverbindung für Ethernetverkabelung

beendet ist, wird dieser Tabelleneintrag von Kolonnenparken-Kontroller wieder auf den boolschen Wert „False“ gesetzt und die LED-Anzeige zeigt ein rotes X an.

3.4.3.1 Ansteuerung über DE/DA oder proprietäreres Protokoll

Aufgrund der vom Kunden eingesetzten Lösung, musste zwischen den oben genannten Kommunikationsformen gewählt werden. Aufgrund der Rahmenanforderung durfte kein proprietäres Protokoll zur Kommunikation mit der LED-Anzeige eingesetzt werden. Der Einsatz eines proprietären Protokolls hätte einen Mehraufwand von Entwicklungsarbeit bedeutet und bei einem möglichen Austausch der Steuerung oder der LED-Anzeigen auf einen anderen Anbieter müsste dieses Protokoll wieder neu entwickelt werden.

Für die Ansteuerung über DE/DA benötigt das AutomationX-System, aber eine zusätzliche Komponente, da es für das Steuerungs- und Visualisierungssystem aufgrund zweierlei Gründe nicht möglich ist:

- Örtliche Distanz, Kabellänge würde enorme Mehrkosten verursachen
- Virtualisierung, die Steuerung ist auf keine Hardware fix gebunden

Für die Lösung des Problems wurde ein kleiner Buskoppler mit 4 integrierten DE und DA vor der LED-Anzeige eingesetzt, der von einem offenen industrielles Kommunikationsprotokoll auf DE/DA umwandelt. Dadurch ist gewährleistet dass die Rahmenanforderungen des Kunden umgesetzt werden können obwohl ein zusätzlicher Materialaufwand nötig ist.

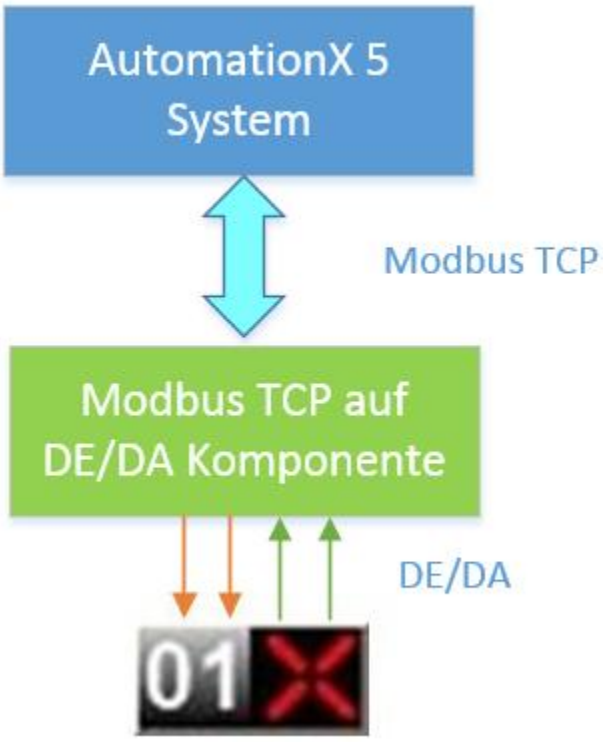


Abbildung 14: Konzept Umsetzung auf DE/DA

3.5 Visualisierung

Da das AutomationX-System auf einer Server-Client Architektur basiert, müssen die verschiedenen Anlagenteile und Prozessbilder nur am Server erstellt werden. Die fertige Applikation wird dann über TCP/IP am Bedien-Rechner oder an den Einfahrtsterminals Touch-Monitore visualisiert. Dies hat den Vorteil, dass keine zusätzliche Software auf den Clients installiert werden muss. Mit einem einfachen Aufruf einer Verknüpfung am Desktop und Eingabe der Login Daten wird die Applikation geladen. Die Informationen sind so immer zentral gespeichert.

Die Grafikobjekte in der Visualisierung können den Zustand der einzelnen Anlagenteile darstellen. Dazu sind diese Objekte je nach aktuellem Zustand entsprechend zu animieren. Die strukturelle und farbliche Darstellung wird mit dem AG abgestimmt, kann aber nachträglich verändert werden.

3.5.1 Parkplatzübersicht

Die Parkplatzübersicht am Bedien-Rechner muss eine Auflösung von 3840x1080 Pixel aufweisen, um den kompletten Parkplatz über 2 FullHD Monitore zu visualisieren. Diese Übersicht ist die Hauptvisualisierung und muss auch als Startbild eingestellt werden.

Folgende Informationen müssen ohne zusätzlichen Funktionsaufruf für den Bediener sichtbar sein:

- Vogelperspektive des Parkplatzes als Hintergrund
- Status Einfahrtsbereich
    - Einfahrt belegt
    - Störung Touch Monitor und Industrie PC im Einfahrtsterminal
    - Schranke offen/geschlossen und Störungen
    - Störungen Tickt Drucker
- LED-Anzeige der Parkstandsreihe
    - Offen/Geschlossen oder Störung
- Parkstandsreihen Information
    - Freie Distanz
    - Status: Gesperrt oder Gebucht
- Fahrzeug Information
    - Position
    - Länge
    - Fahrzeugart
    - Wildparker
- Legende für die Farbgebung

Damit diese Informationen dem Bediener immer in aktualisierter Form zur Verfügung stehen wird eine zyklische Abfrage auf den unten angeführten Tabellen ausgeführt, um die Daten aus der Datenbank auszulesen:

- ReservedParkingQueue
- VehicleData
- ParkingQueue

3.5.2 Konfigurations-Parameter

Das intelligente Kolonnenparksystem lässt sich frei konfigurieren. Die Daten sind in der zentralen Datenbank abzuspeichern. Während des Systemstarts werden sämtliche, die Anlage konfigurierenden Daten, sowie sämtliche zum Betrieb notwendigen Parameter aus der zentralen Datenbank ausgelesen. Alle Konfigurations-Parameter können bei Bedarf Online geändert und sofort nach Bestätigung übernommen werden. Änderungen in der Parametrierung werden ohne Neustart im nächsten Zyklus übernommen. Konfigurationsänderungen können je nach Art der Änderung (z.B. Änderung der Konfiguration für die Zwölfstundenregel) einen Neustart des Systems erforderlich machen. Sämtliche Konfigurationen und Parametrierungen werden versioniert und archiviert, sodass bei Bedarf auf einen alten Konfigurationsstand und Parametersatz zurückgegriffen werden kann.

Fehleingaben des Operators können nicht zu Systemausfällen führen. Alle numerischen Eingaben sind mit Hilfe von Grenzwerten einzutragen. Auf Fehlbedienung wird der Bediener wird mit aussagekräftigen Fehlermeldungen aufmerksam gemacht.

3.5.3 Alarme

Das AutomationX-System verfügt über eine prioritätengestützte Alarmierungsfunktion. Diese ist im System fest definiert und kann nicht ohne weiteres verändert werden. Die Alarme werden an den Bedien-Rechnern visuell dargestellt. Tritt eine Störung oder ein Alarm an einem Anlagenteil auf, so wird dies am entsprechenden Grafikobjekt und aller hierarchisch darüber befindlichen Grafikobjekte durchgängig dargestellt.

Sowohl das Auftreten als auch das Quittieren der Alarme wird protokolliert und steht für Auswertungen zur Verfügung.

Die Alarmierung wird in 4 farbliche Gruppen eingeteilt um den Bediener schnell sichtbar zu machen, welche Anlagenteile einen sofortigen Technikereinsatz benötigen.

- **Priorität 1-10 Meldung: Hellblau**
- **Priorität 11-19 Störung oder Warnung: Gelb**
- **Priorität 20-29 Alarme: Rot**
- **Priorität >100 AutomationX-Systemalarme: Grau**

4 Implementierung

Im vorangegangenen Abschnitt wurden die zu übertragenden Daten zwischen den Systemen dargestellt. In diesem Kapitel werden der Aufbau und die Umsetzung der Kommunikation zwischen den Systemen erläutert. Da AutomationX-Software die eigenentwickelte Software der AutomationX GmbH ist, und die Gegenstelle wenn möglich in Zukunft beliebig ersetzbar sein sollte, liegt der Fokus der Betrachtung auf den zu implementierenden Funktionen des Steuerungs- und Visualisierungssystems. Diese werden in diesem Kapitel detailliert dargestellt.

4.1 Erstellen der Schnittstellen

In diesem Unterkapitel werden die Implementierungen der einzelnen Schnittstellen näher beschrieben um die geforderten Funktionen abzubilden. Die AutomationX-Software ist objektorientiert aufgebaut, deshalb werden im folgenden Unterkapitel mehrmals die objektorientierten Begriffe für Klassen, Variablen und Instanzen genannt.

4.1.1 Aktivieren der REST Schnittstelle

Da das AutomationX-System eine integrierte REST Schnittstelle besitzt kann auf diese direkt zugegriffen werden. Für jeden Endpunkt muss eine eigene Instanz angelegt und die dazugehörigen Methoden manuell eingetragen werden. Nach dieser Konfiguration sind die Endpunkte im ganzen AutomationX-System verfügbar und können zyklisch aufgerufen werden.

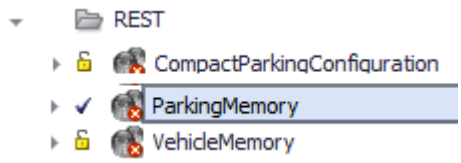


Abbildung 15: REST Instanzen

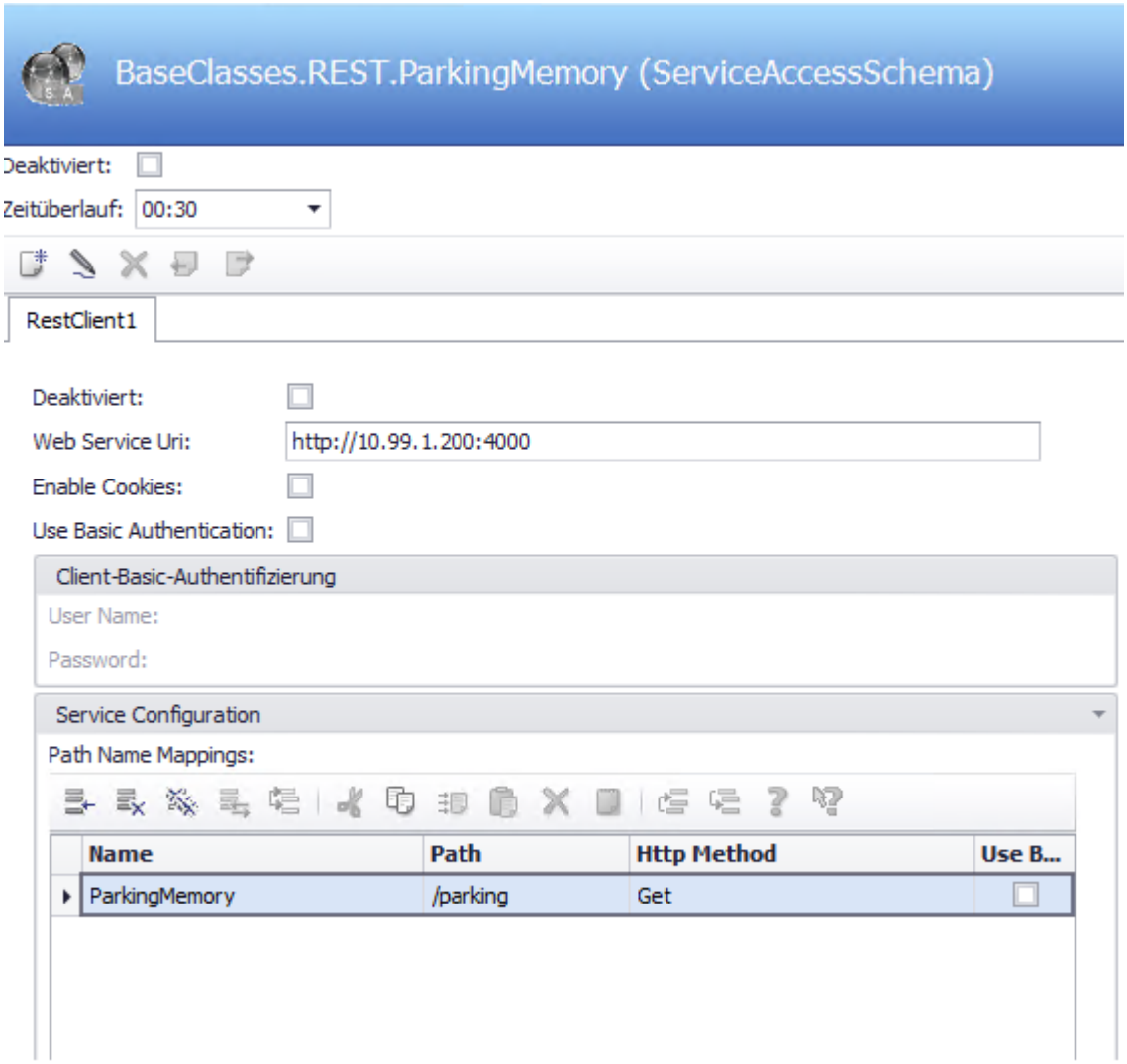


Abbildung 16: REST Integration Parkplatzinformation

Weiterführende Informationen zur Konfiguration im AutomationX-System gibt es unter [REST2018].

4.1.2 Aktivieren der Datenbankschnittstelle

Das SqlSchema definiert eine SQL-Datenquelle. Derzeit werden folgende Gerätetypen unterstützt:

- MSSQL (Microsoft SQL Server)
- ODBC (eingeschränkt)
- TFS (Microsoft TeamFoundationServer)

Mit den integrierten Tools kann, wie in der nachfolgenden Abbildung dargestellt, die MS SQL 2016 Datenbank „ParkingLotData“ mit den AutomationX-System verknüpft werden und dann einzelne T SQL Statements abgesetzt und gespeichert werden.

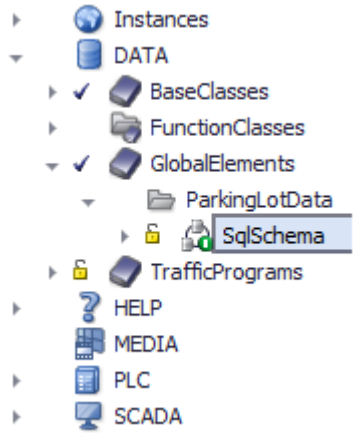


Abbildung 17: SQL Instanz

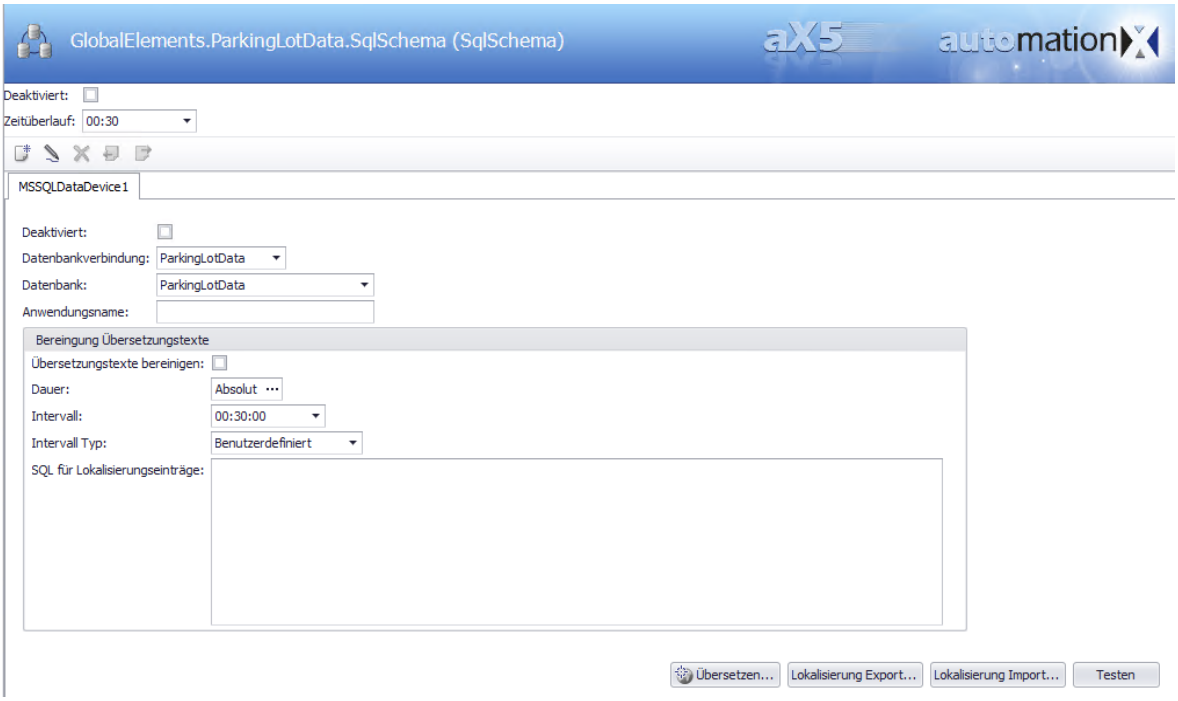


Abbildung 18: Datenbank Schema einfügen

Auf die „Always On Avaibility Cluster“ Redundanz der Datenbank muss keine Rücksicht genommen werden. Das AutomationX-System benötigt nur die Cluster IP-Adresse der Redundanzeinstellungen als Gegenstelle und der eingestellte Windows Cluster verwaltet automatisch zwischen Master- und Slavebetrieb.

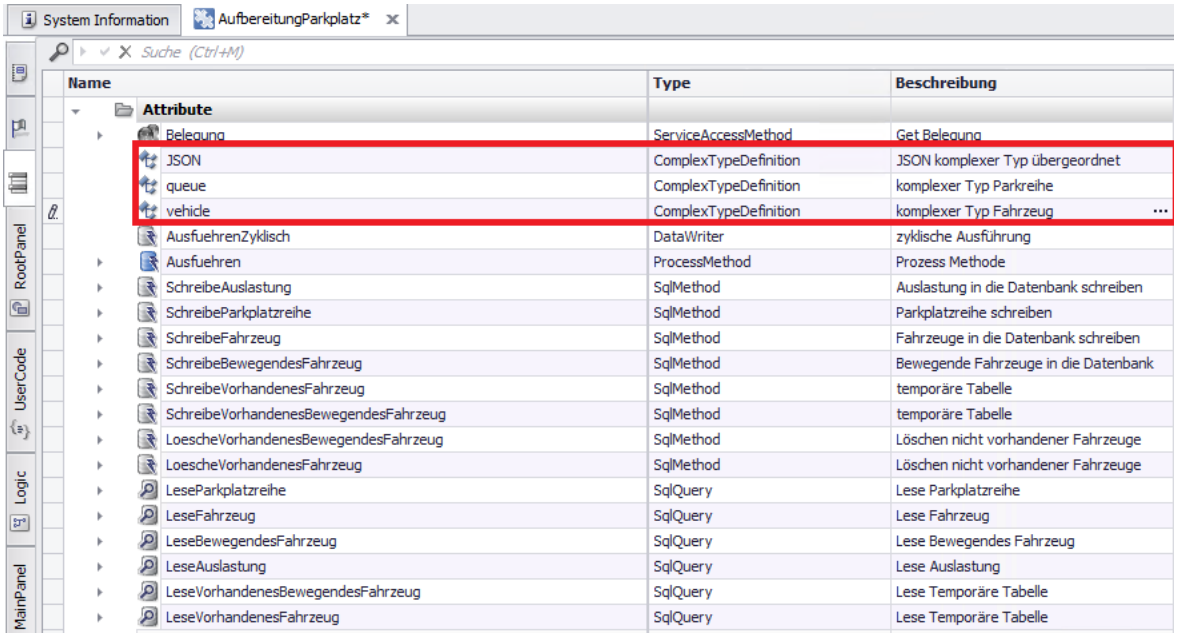
Weiterführende Informationen zur Verwendung des MIB Browser gibt es unter [SQL2018].

4.1.3 Abbilden der JSON Formate

Das AutomationX-System bietet eine integrierte Funktion die das komplette JSON Format, wie in Kapitel 3.1 definiert, inklusive aller Objekte, Arrays und verschiedenen Datentypen.

Dieses Abbild des JSON Formats im AutomationX-System dient dazu, die Daten korrekt von der REST Schnittstelle zu empfangen und über das AutomationX-System in die Datenbank einzutragen. Dieses Abbild funktioniert auch um z.B. POST Methoden zum REST Server zu senden. Hierfür werden Daten aus der Datenbank ausgelesen und dann vom AutomationX-System in ein JSON Format für die REST Schnittstelle umgewandelt.

Die Verschachtelungen funktionieren im AutomationX-System mit Komplexen Typen, die gleichbedeutend mit den JSON Objekten sind.



Name	Type	Beschreibung
Attribute		
Belegung	ServiceAccessMethod	Get Belegung
JSON	ComplexTypeDefinition	JSON komplexer Typ übergeordnet
queue	ComplexTypeDefinition	komplexer Typ Parkreihe
vehicle	ComplexTypeDefinition	komplexer Typ Fahrzeug
AusfuehrenZyklisch	DataWriter	zyklische Ausführung
Ausfuehren	ProcessMethod	Prozess Methode
SchreibeAuslastung	SqlMethod	Auslastung in die Datenbank schreiben
SchreibeParkplatzreihe	SqlMethod	Parkplatzreihe schreiben
SchreibeFahrzeug	SqlMethod	Fahrzeuge in die Datenbank schreiben
SchreibeBewegendesFahrzeug	SqlMethod	Bewegende Fahrzeuge in die Datenbank
SchreibeVorhandenesFahrzeug	SqlMethod	temporäre Tabelle
SchreibeVorhandenesBewegendesFahrzeug	SqlMethod	temporäre Tabelle
LoescheVorhandenesBewegendesFahrzeug	SqlMethod	Löschen nicht vorhandener Fahrzeuge
LoescheVorhandenesFahrzeug	SqlMethod	Löschen nicht vorhandener Fahrzeuge
LeseParkplatzreihe	SqlQuery	Lese Parkplatzreihe
LeseFahrzeug	SqlQuery	Lese Fahrzeug
LeseBewegendesFahrzeug	SqlQuery	Lese Bewegendes Fahrzeug
LeseAuslastung	SqlQuery	Lese Auslastung
LeseVorhandenesBewegendesFahrzeug	SqlQuery	Lese Temporäre Tabelle
LeseVorhandenesFahrzeug	SqlQuery	Lese Temporäre Tabelle

Abbildung 19: JSON Objekte anlegen

Die Objekte müssen mit den einzelnen Feldnamen und Datentypen erstellt werden. Falls es sich um ein Array handelt muss der Optionale Parameter „Array“ aktiviert werden.

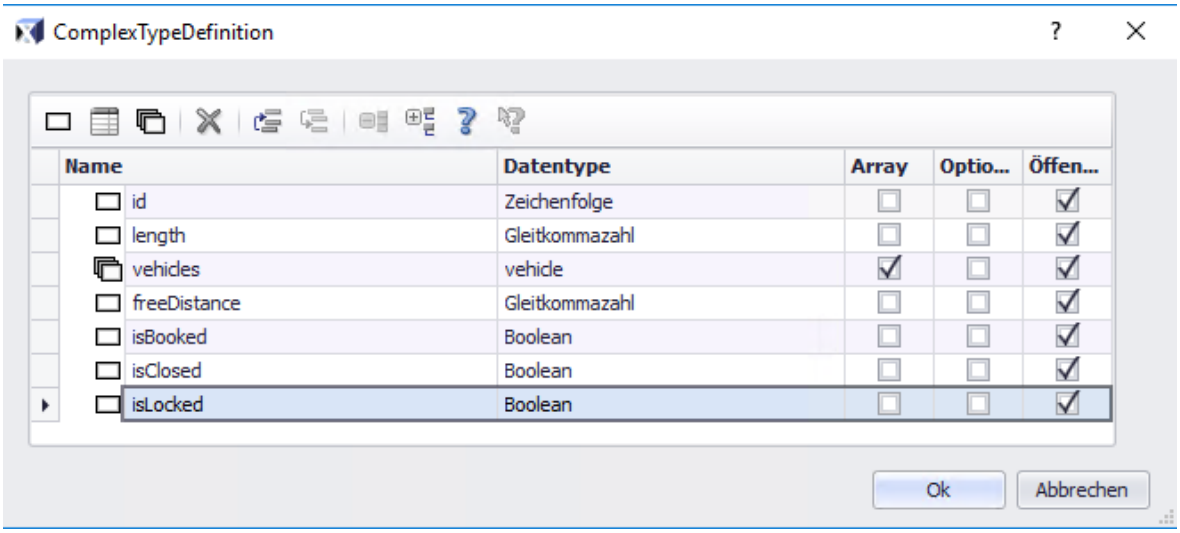


Abbildung 20: JSON Feldnamen anlegen

Weiterführende Informationen zur Verwendung des MIB Browser gibt es unter [JSON2018]

4.2 Erstellen der Steuerung

Nachdem die Schnittstellen zum Kolonnenparken-Kontroller und zur Datenbank erstellt wurden, kann mit der Erstellung der übergeordneten Funktionsweise begonnen werden.

Mit übergeordneter Funktionsweise sind die zusammenhängenden Abläufe der verschiedenen Anlagenteile gemeint.

4.2.1 Anlegen der Variablen

Damit das AutomationX-System über Kommunikationsprotokolle mit anderen Systemen Daten austauscht, müssen erst Variablen erstellt werden. So kann eine Verknüpfung zwischen Datenbank, Feldbus oder Kolonnenparken-Kontroller hergestellt werden.

Variablen können steuerungsübergreifend verwendet werden. Je nach Systemaufbau wird die SPS entweder am lokalen Steuerungs- und Visualisierungssystem oder abgesetzt auf einem Industrie PC verwendet. Für die Implementierung müssen für jede zu erstellende Klasse die dazugehörigen Variablen mit den richtigen Datentyp angelegt werden. Die Datentypen entsprechen der IEC 61131-3 Norm. Zusätzlich gibt es aber auch einen spezifischen Datentyp des AutomationX-Systems der Datentyp „Bool“ ähnelt, aber zusätzliche Alarmierungs-Attribute enthält und dadurch eine automatisches Visualisieren von Alarmzuständen und automatisches Speichern des Ereignisses in der Datenbank erleichtert.

Die meisten Variablen sind schon durch die jeweiligen Definitionen der einzelne Schnittstellen und Funktionen gegeben und können übernommen werden.

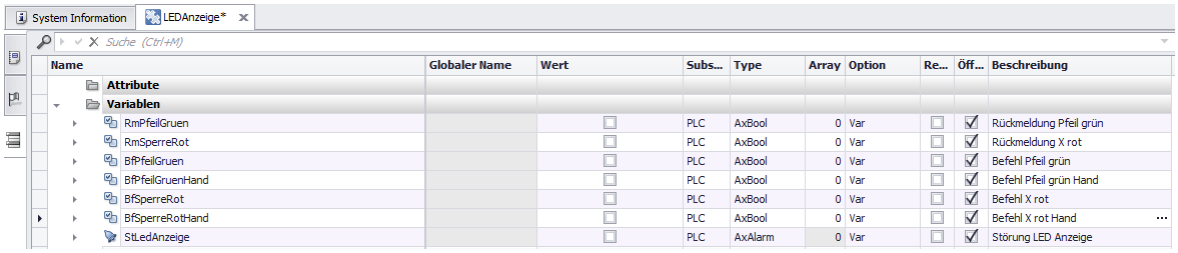


Abbildung 21: Anlegen Variablen

Weiterführende Informationen zur Konfiguration der SNMP Schnittstelle im AutomationX-System gibt es unter [Variablen2018].

4.2.2 Erstellen der Feldbus-Kommunikation

Zuerst musste vom Schranken und der LED-Anzeige die jeweilige Datenpunktliste oder Systembeschreibung von den jeweiligen Lieferanten angefordert werden, um dann mit dem AutomationX-System einen Kommunikations-Datenpunkt und eine AutomationX Variable miteinander verknüpfen zu können. Anhand der in den oberen Kapiteln genannten Meldungen, Störungen und Befehle müssen die gewünschten Datenpunkte gesucht werden. Die Kommunikations-Datenpunkte sind bei Modbus TCP in Busadresse und Offsets dargestellt.

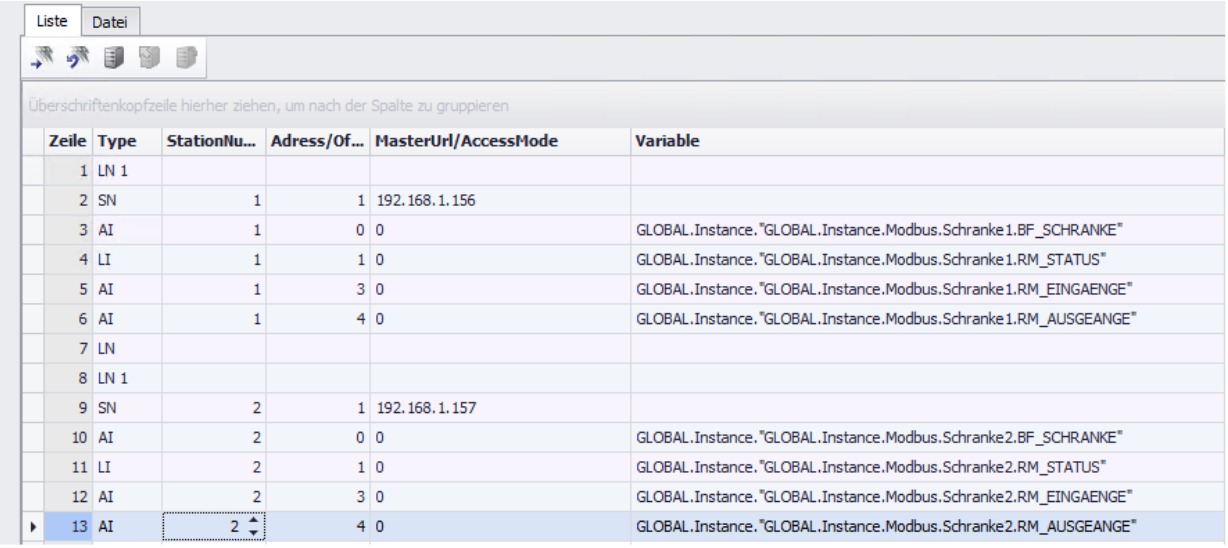
Adresse	Name	Daten-typ	Be-reich	Zugriff	Beschreibung	Objekt-Mapping Knoten/Index/Subindex
0000	Bar-Control	16Bit	0 … 4	RW	Schrankenbefehle: 0000 – Befehle löschen 0001 – öffnen 0002 – schließen 0004 [1] – übergeordnet öffnen	
0001	Status	32Bit		RO	Statuswort der Schranke	Logic Controller/ 0x1002/0x00
0003	Inputs	16Bit		RO	Zustände der Eingänge der MGC-PRO-Steuergerät	Logic Controller/ Bit0 – 7 – 0x6000/0x01 Logic Controller/ Bit8 –15 – 0x6000/0x02
0004	Out-puts	16Bit		RO	Zustände der Ausgänge der MGC-PRO-Steuergerät	Logic Controller/ 0x6300/0x01

Abbildung 22: Modbus Datenpunktsliste Schranke aus der Quelle [Schranke2018]

Nachdem alle Modbus Datenpunkte mit den jeweiligen Adressen herausgesucht und getestet wurden muss diese Information in das AutomationX-System, in einer gewissen Formatierung, geladen werden. Das AutomationX-System benötigt folgende Informationen in Listenform:

- IP-Adresse der Gegenstelle
- Datenpunktadresse
- Datentyp
- Schranken- und LED-Anzeige-Instanz

In der nachfolgenden Abbildung wird die Formatierung der Liste mit den zusätzlich benötigten Daten dargestellt.



Liste | Datei

Überschriftenkopfzeile hierher ziehen, um nach der Spalte zu gruppieren

Zeile	Type	StationNu...	Adress/Of...	MasterUrl/AccessMode	Variable
1	LN 1				
2	SN	1	1	192.168.1.156	
3	AI	1	0	0	GLOBAL.Instance."GLOBAL.Instance.Modbus.Schranke1.BF_SCHRANKE"
4	LI	1	1	0	GLOBAL.Instance."GLOBAL.Instance.Modbus.Schranke1.RM_STATUS"
5	AI	1	3	0	GLOBAL.Instance."GLOBAL.Instance.Modbus.Schranke1.RM_EINGAENGE"
6	AI	1	4	0	GLOBAL.Instance."GLOBAL.Instance.Modbus.Schranke1.RM_AUSGEANGE"
7	LN				
8	LN 1				
9	SN	2	1	192.168.1.157	
10	AI	2	0	0	GLOBAL.Instance."GLOBAL.Instance.Modbus.Schranke2.BF_SCHRANKE"
11	LI	2	1	0	GLOBAL.Instance."GLOBAL.Instance.Modbus.Schranke2.RM_STATUS"
12	AI	2	3	0	GLOBAL.Instance."GLOBAL.Instance.Modbus.Schranke2.RM_EINGAENGE"
13	AI	2	4	0	GLOBAL.Instance."GLOBAL.Instance.Modbus.Schranke2.RM_AUSGEANGE"

Abbildung 23: Auszug AutomationX Modbusliste

Weiterführende Informationen zur Konfiguration der SNMP Schnittstelle im AutomationX-System gibt es unter [ModbusTCP2018].

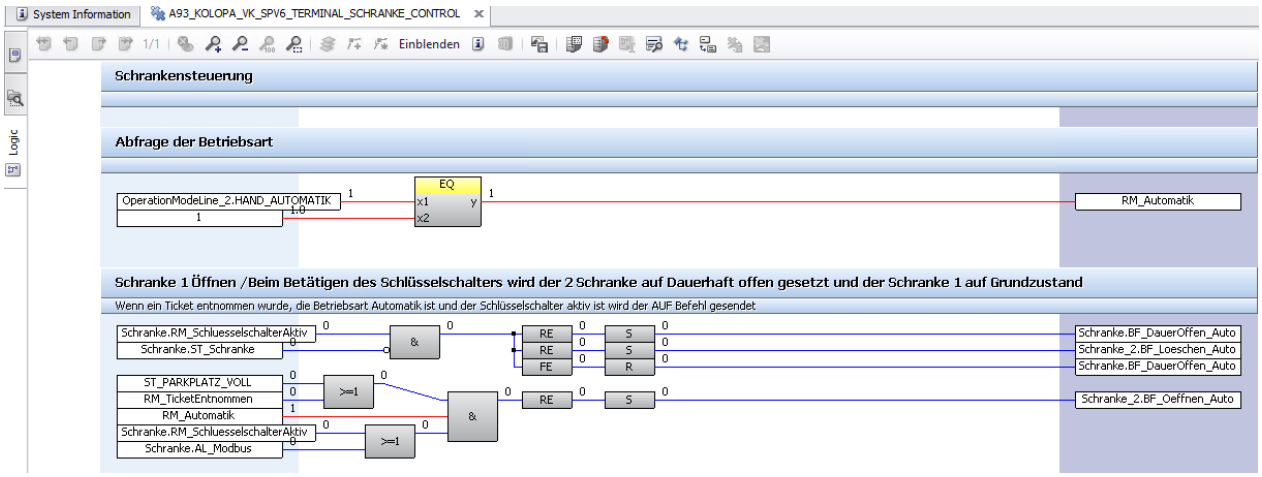
4.2.3 Erstellen der Funktionen

Die Funktion wird in mit dem AutomationX SPS-Logik Editor erstellt und es kann als neues Logik Element zu einer Klasse hinzugefügt werden. Das Abbilden der Funktion wird in FUP[34] realisiert.

Eine Funktion besteht aus einer oder mehrerer Logikseiten (Networks), die zur Gliederung und besseren Übersichtlichkeit von graphischen SPS-Codes dienen und welche von der SPS sequentiell abgearbeitet werden. Beim Erzeugen einer neuen Logik unter einer Klasse wird standardmäßig eine leere Logikseite erzeugt. Sobald eine Klasse über eine Logik verfügt, können Instanzen dieser Klasse auf SPS Geräte gebunden und von diesen exekutiert werden.

Die zuvor definierten Abläufe z.B. für die Schrankensteuerung oder LED-Anzeige werden dann in den oben genannten Logikseiten erstellt und können auf den jeweiligen verteilten Systemen exekutiert werden.

[34] FUP ist eine SPS Programmiersprache (siehe https://www.sps-lehrgang.de/funktionsplan-fup/)



Weiterführende Informationen zur Konfiguration der SNMP Schnittstelle im AutomationX-System gibt es unter [Logik2018].

4.3 Erstellen der Visualisierung

Das AutomationX-System verfügt über einen integrierten Grafik-Editor indem einzelne grafische Objekte als Klassen gespeichert werden können. Die Visualisierungsbilder werden in Automatisierungsklassen abgebildet, deren Erstellung im Systemkonzept beschrieben ist. In den Basisobjekten der grafischen Darstellung werden visuelle Effekte und Aussehen definiert.

Weiterführende Informationen zur Konfiguration der grafischen Oberfläche und Objekten im AutomationX-System gibt es unter [Graphic2018].

4.3.1 Übersichtsbild

Die einfachste Form eines Prozessbildes ist die Darstellung über die gesamte Größe. Dabei sind alle Informationen reine Visualisierungen einer Anlage. Hierfür wird in einer Klasse das RootPanel verwendet. Dieses stellt im Grunde die Zeichenfläche, also den Startpunkt für das Erstellen eines Prozessbildes, dar. Hierauf können dann andere Klassen oder Instanzen platziert werden. Dies funktioniert entweder aus dem Klassenfenster, Projektfenster oder aus dem Objektexplorer heraus. Das Übersichtsbild ist letztendlich eine Summe aus verschiedenen Visualisierungen von Klassen und Instanzen über viele Ebenen.

Auf das Übersichtsbild werden folgende Instanzen platziert:

- Einfahrtsterminal
- Schranken
- LED-Anzeige 1-31
- Status der Parkstandsreihe
- Freie Distanz der Parkstandreihe
- Fahrzeuge auf den Parkstandsreihen

- Kameras
- Legende

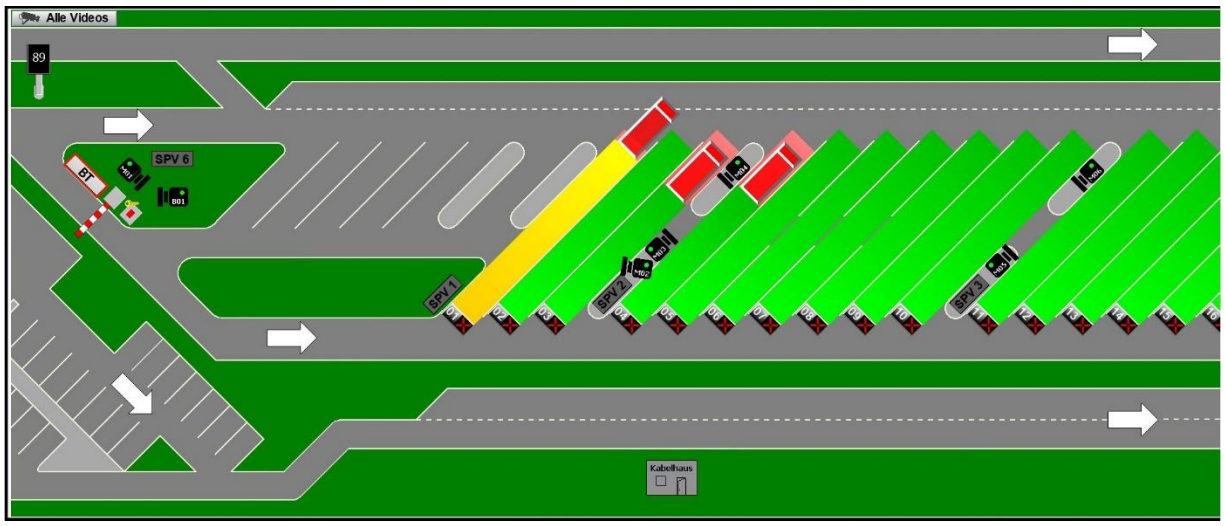


Abbildung 24: Übersichtsbild Teil 1

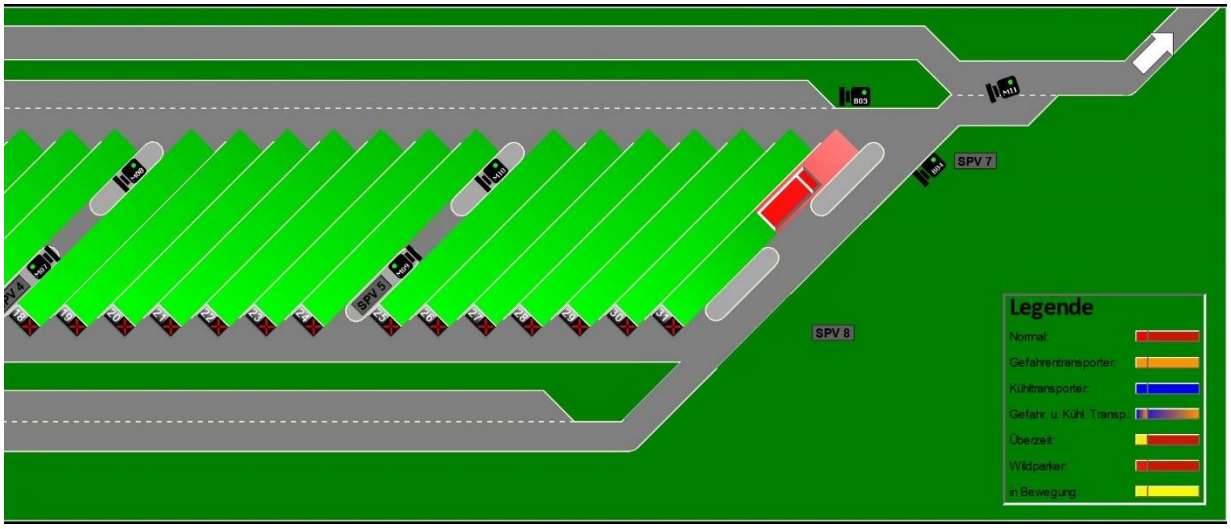


Abbildung 25: Übersichtsbild Teil 2

Der Bediener bekommt durch die oben genannten positionierten Objekte einen schnellen Überblick über den Status des Parkplatzes und kann gegebenenfalls bei Fehlfunktionen eingreifen.

4.3.2 Erstellen der Visualisierungsobjekte

Dazu wird eine neue Klasse im AutomationX-System mit dem Namen „Schranke“ und „LEDAnzeige“ erstellt. In diesen Klassen sind dann die jeweiligen Grafikobjekte, Befehle, Alarme und Variablen für die diversen Funktionen integriert. Die Klassen können dann mehrmals im Projekt instanziert werden und stehen dann der Visualisierung zur Verfügung. Eingriffe und Schalthandlungen erfolgen dagegen über Panels. Dies sind eigene kleine Fenster, welche sich zum Beispiel mit Schaltflächen verknüpfen lassen. Wählt der Bediener eine derart verknüpfte Schaltfläche aus, so öffnet sich das Panel. Der Inhalt des Panels wird gleich wie ein Prozessbild zusammengestellt. Die Größe eines Panels ist da-

bei nicht limitiert, sollte aber eher klein gehalten werden. Aus jedem Panel können weitere Panels geöffnet werden. Des Weiteren wird die Position eines Panels auf den jeweiligen Bildschirmen gespeichert, sodass es nach Schließen und erneutem Öffnen wieder an der gleichen Stelle erscheint. Jedem geöffneten Panel wird zudem das vorhergehende Element zugeordnet, durch welches es geöffnet wurde. Befindet sich der Mauszeiger über einem Panel, so wird das zugehörige Element auf dem Prozessbild optisch hervorgehoben. Dies erleichtert gerade auf Bedienplätzen mit mehreren Bildschirmen die Orientierung.

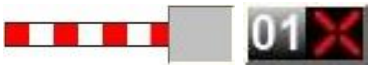


Abbildung 26: Objekte im Prozessbild

Abbildung 27: Panel der Objekte

In diesen Panels, die mit einem Links-Klick geöffnet werden könne, sind folgende Funktionen der „Schranke“ oder „LEDAnzeige“ hinterlegt:

- Instanzname
- Beschreibung
- Aktueller Status
- Betriebsart
    - Handbetrieb
    - Automatik
- Befehle
- Meldungen

- Alarme

Sollte das jeweilige Betriebsmittel in Handbetrieb oder der Bediener keine Autorisierungsstufe besitzen, sind die Befehlsfunktionen gesperrt.

4.3.3 Objekt Konfigurations-Parameter

Dem Bediener muss die Möglichkeit geboten werden, nötige Konfigurations-Parameter umzustellen. Diese Konfigurations-Parameter dienen der Änderung der Detektion oder des zeitliche Verhaltens.

Diesbezüglich wird dann ein grafisches Objekt mit dem Namen „Parameter“ erstellt. Wichtig sind diverse Bedienereinschränkungen für gewisse Parameter, weil nicht jeder Bediener alle Parameter verändern darf. Das AutomationX-System stellt für diese gewünschten Einschränkungen 4 Authentifizierungsgruppen zur Verfügung:

- Administrator: Kann alle Parameter ändern und einzelne Felder sperren
- Konfigurator: Kann alle Parameter ändern aber keine Felder sperren
- Operator: Kann nur Parameter ändern die der Administrator freigegeben hat
- Gast: Kann Parameter nur ansehen und keine Änderungen durchführen

Bei der Implementierung der Konfigurations-Parameter muss jedes Eingabefeld mit einem Wahrheitswert geprüft werden um Falscheingaben zu verhindern. Dazu können einzelne Eingabefelder mit den Bedingungen verknüpft werden und mit einer Abbruch-Funktion versehen werden.

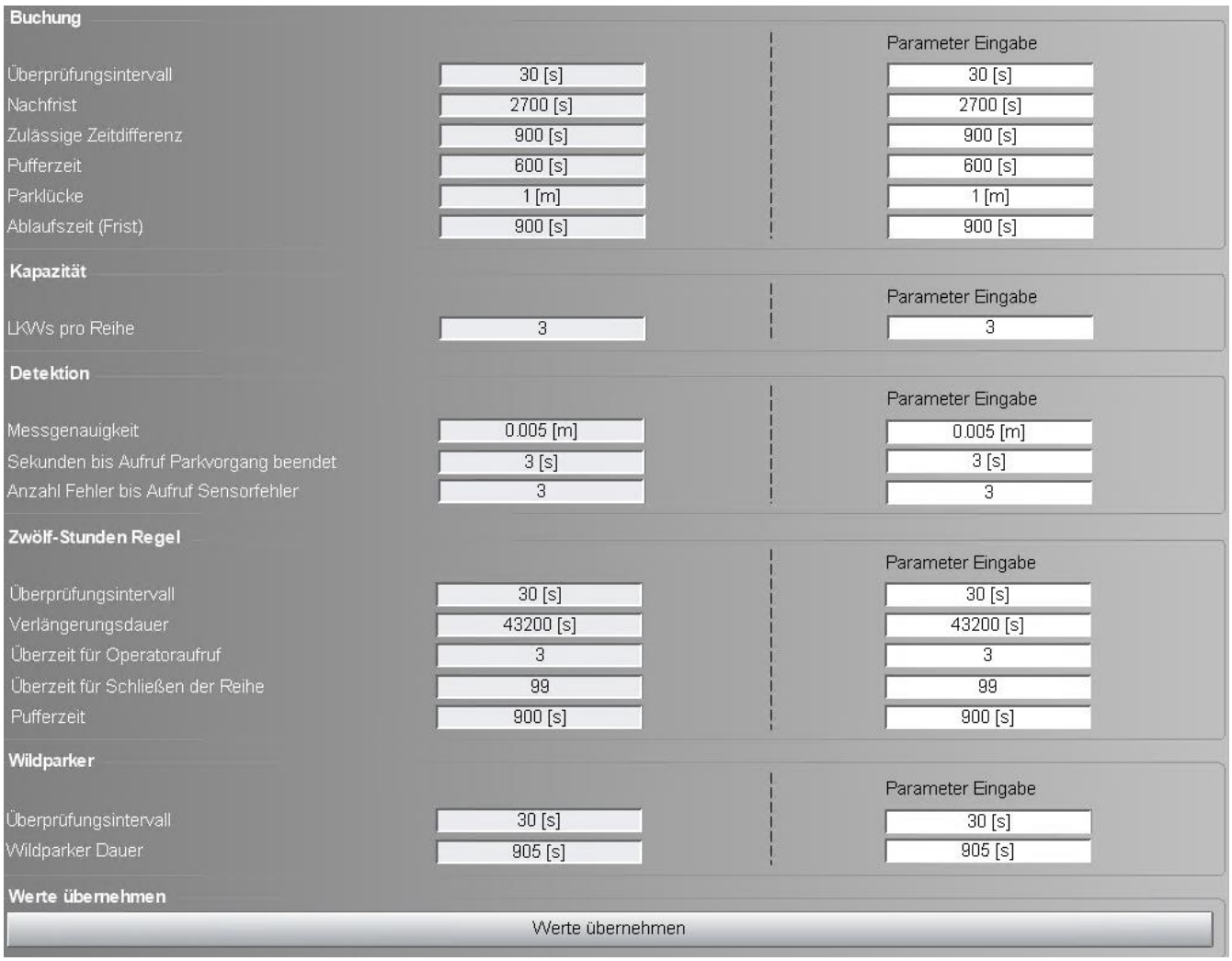


Abbildung 28: Konfigurations-Parameter

Auf der linken Seite des Konfigurations-Parameter-Objekts sind die einzelnen Einträge in Untergruppen eingeteilt, um eine bessere Übersicht zu bekommen. Bei den linken Parameter Einträgen, die hellgrau hinterlegt sind, handelt es sich um Istwerte. Das bedeutet, dass diese Werte nicht änderbar sind und nur den aktuellen Systemstatus wiedergeben.

Auf der rechten Seite können, je nach Autorisierung, Werte verändert werden. Nachdem alle gewünschten Werte eingetragen sind, können mit dem „Werte übernehmen“ Knopf, die Werte in JSON Format über die REST Methode „POST /config“ zum Kolonnenparksystem übertragen werden.

5 Ergebnisse und Ausblick

Im letzten Kapitel werden die Ergebnisse zusammengefasst, ein kurzer Ausblick gegeben und die Umsetzungsmethode definiert.

5.1 Ergebnisse

Das Thema für die Diplomarbeit " Schnittstelle, Steuerung und GUI für ein LKW-Kolonnenparksystem" ergab sich aufgrund eines Kundenauftrages von der südbayerischen Autobahndirektion und der Firma Neurosoft, die den Kolonnenparken-Kontroller liefert. Mit diesen beiden Kunden wurde erstmals ein Projekt mit unserem hauseigenen Produkt durchgeführt. Der intelligente LKW Kolonnenparkplatz war technisches Neuland für alle drei Parteien in diesem Projekt. Gerade deshalb war es wichtig einen Weitblick für die Zukunft zu haben, da die Möglichkeit besteht diesen Ansatz eines intelligenten LKW-Kolonnenparkens auf Parkplätze in Deutschland oder ganz Europa einzusetzen. Das heißt die Schnittstellen der verschiedenen Anlagenteile sollten, wenn es möglich ist, auf offene Industriestandards ausgelegt sein, weil diese sich nicht so schnell verändern wie die Hardware oder Software.

Am Anfang war es deshalb für das Systemkonzept und die Anforderungen des Kunden essentiell eine möglichst offene und effiziente Lösung zu finden. Aufgrund der integrierten Systemfunktionen des AutomationX-Systems und der relativ großen Auswahl von freien Bibliotheken, konnten mit den Kunden relativ schnell einzelne Schnittstellen, Abläufe und die Visualisierung definiert werden.

Leider musste bei den vom Kunden gelieferten LED-Anzeigen eine zusätzliche Komponente verbaut werden, um diese den Rahmenanforderungen entsprechend anzusteuern. Im Endeffekt war der Kunde aber mit der vorgeschlagenen Lösung zufrieden und das Preis-Leistungs-Verhältnis war für den Kunden positiv.

Durch das AutomationX Produkt konnten nach dem abgestimmten Systemkonzept, aber alle Funktionen mit bestehenden Werkzeugen und Funktionen ohne große Komplikationen integriert werden. Ein Problem, dass sich am Ende bei den Tests herausstellte war kurzeitig die hohe CPU Last auf dem Steuerungs- und Visualisierungssystem, weil die Abfrage und Schreibzyklen in der Datenbank zu kurz waren. Dieses Problem konnte aber schnell gelöst werden in dem die Abfragezyklen geändert und verteil wurden, damit nicht alles gleichzeitig gelesen und geschrieben wird.

Das Endergebnis für das Steuerungs- und Visualisierungssystem war für den Kunden und für AutomationX positiv. Trotz einige technische Schwierigkeiten und Probleme die bei der Neuentwicklung aufgetreten sind, ist im Endeffekt aus einem Projekt ein Produkt gewor-

den, das zusammen mit der Firma Neurosoft auch auf anderen Parkplätzen eingesetzt werden kann. Wodurch ein Wettbewerbsvorteil gegenüber Mitbewerber entstanden ist.

5.2 Bewertung der Arbeit

Für diese Arbeit war eine Einarbeitung in ein breites Themengebiet notwendig. Nur durch gutes technisches Basiswissen aus den Projektgegebenheiten konnten Zusammenhänge schnell und vollständig erfasst werden, um Schlüsse daraus zu ziehen und dementsprechende Lösungen umzusetzen.

Aufgrund der engen Zusammenarbeit mit dem Kunden und der sehr offen Kommunikation, die man über die Projektphase aufgebaut hat, konnten schon im Vorhinein die einzelnen Wunschfunktionen, Protokollaufbau und Standards von unserer, wie auch von Seiten des Kunden gut abgestimmt werden. Der Kunde hatte nicht immer den Blick für das große Ganze um das Softwareprodukt auch für mögliche zukünftige Ausrollphasen vorzubereiten. Deshalb brauchte der Kunde zusätzliches Know How, das in der Firma AutomationX im Infrastruktur Bereich vorhanden ist.

Wichtig ist, dass man trotz der geographischen Distanz des Kunden und AutomationX, nicht immer alles über Telefon, Videokonferenzen und E-Mail abklärt, sondern sich auch persönlich trifft. Dadurch kann einiges an Zeit und falschen Interpretationen eingespart werden. Am Anfang des Projekts kam es deshalb zu vielen Nachbesserungen für den Datenaustausch und zu Änderungen des Systemaufbaus.

Am Ende ist es aber trotz der Neuentwicklung von der Firma AutomationX und Neurosoft gelungen, dem Kunden ein Produkt zu liefern, das alle neuen technischen Anforderungen erfüllt. Die Bediener oder Fahrer waren sofort mit den Funktionen vertraut, ohne größeren Schulungsbedarfs.

5.3 Ausblick

Durch den steigenden LKW-Warenverkehr in Europa wird es zu erhöhten Platz- und Raumproblemen auf Autobahnparkplätzen kommen. Damit alle LKW-Fahrer die restriktiven gesetzlichen Ruhezeiten einhalten können, muss der Autobahnbetreiber die nötige Infrastruktur zur Verfügung stellen. Im Bereich der allgemeinen Parkraumbewirtschaftungen auf Autobahnen wird sich deshalb in den nächsten 10 Jahren noch viel ändern und es wird voraussichtlich versucht auch die Vereinheitlichung der Standards und des Konzepts in der Europäischen Union für das intelligente LKW-Kolonnenparken durchzusetzen. Das Konzept des intelligenten LKW-Kolonnenparkens ist schon ein Anfang, aber dennoch gibt es verschiedene Varianten um eine höhere Parkplatzdichte zu erreichen als bisher und manche Autobahnbetreiber in Europa stehen dem Konzept sehr kritisch gegenüber. Wichtig ist aber, dass der Kunde nun für den zukünftigen Betrieb die neuste Technik und die neusten Funktionen in das System integriert hat und es werden sehr viele verschiedene Autobahnbetreiber, die tatsächlich erreichte Steigerung der Parkplatzfläche beobachten. Wird der Parkplatz von den Fahrern akzeptiert und die eingesetzte Technik erweist sich

als robust und wenig fehleranfällig, wird mit der Firma Neurosoft wahrscheinlich eine Partnerschaft abgeschlossen, um dieses Produkt zusammen in künftigen Ausschreibungen anzubieten. Der Wettbewerbsvorteil und die Erfahrungswerte werden einen großen Preis-Leistungs-Unterschied zwischen uns und den Mitbewerbern zur Folge haben.

Literatur

[Aufbau2018] Neurosoft
URL:<https://confluence.neurosoft.pl/pages/viewpage.action?pageId=110723134>
verfügbar am 13.07.2018

[bmvi2018] Bundesministerium
URL:<https://www.bmvi.de/SharedDocs/DE/Artikel/StB/nebenbetriebe-rastanlagen.html> >
verfügbar am 13.07.2018

[RESTAPI2018] CloudComputing
URL:<https://www.cloudcomputing-insider.de/was-ist-eine-rest-api-a-611116/>
verfügbar am 21.07.2018

[Methoden2018] Thomas Bayer
URL:< https://www.oio.de/public/xml/rest-webservices.htm>
verfügbar am 13.07.2018

[DokuKolopa2018] AutomationX
Kolopa_Beschreibung_AX_V1.0
2018

[REST2018] AutomationX
URL:<https://xwiki.automationx.com/wiki/axdocumentation/view/aX5/Basics/Library/Communication Classes/Test/>
verfügbar am 13.07.2018

[SQL2018] AutomationX
URL:<https://xwiki.automationx.com/wiki/axdocumentation/view/aX5/Basics/Library/Communication Classes/Test/>
verfügbar am 13.07.2018

[JSON2018] AutomationX
URL:<https://xwiki.automationx.com/wiki/axdocumentation/view/aX5/Basics/Library/Communication Classes/Test/>
verfügbar am 13.07.2018

[Variablen2018] AutomationX
URL:<https://xwiki.automationx.com/wiki/axdocumentation/view/aX5/Basics/Library/Communication Classes/Test/>
verfügbar am 13.07.2018

[Schranke2018] Magnetic Autocontrol
58150001DE_00_Technisches Handbuch EM01
2017, S. 18

[ModbusTCP2018] AutomationX
URL:<https://xwiki.automationx.com/wiki/axdocumentation/view/aX5/Basics/Library/Communication Classes/Test/>
verfügbar am 13.07.2018

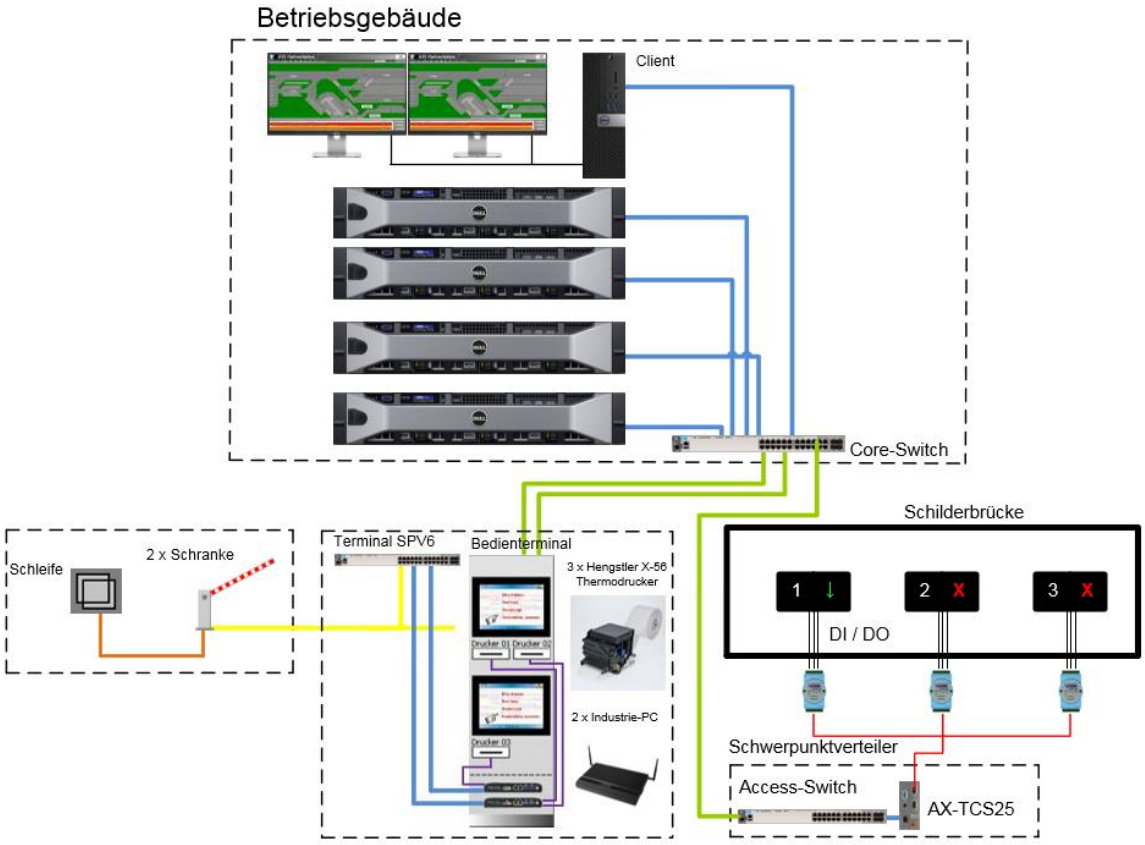
[Logik2018] AutomationX
URL:<https://xwiki.automationx.com/wiki/axdocumentation/view/aX5/Basics/Library/Communication Classes/Test/>
verfügbar am 13.07.2018

[Graphic2018] AutomationX
URL:<https://xwiki.automationx.com/wiki/axdocumentation/view/aX5/Basics/Library/Communication Classes/Test/>
verfügbar am 13.07.2018

Anlagen

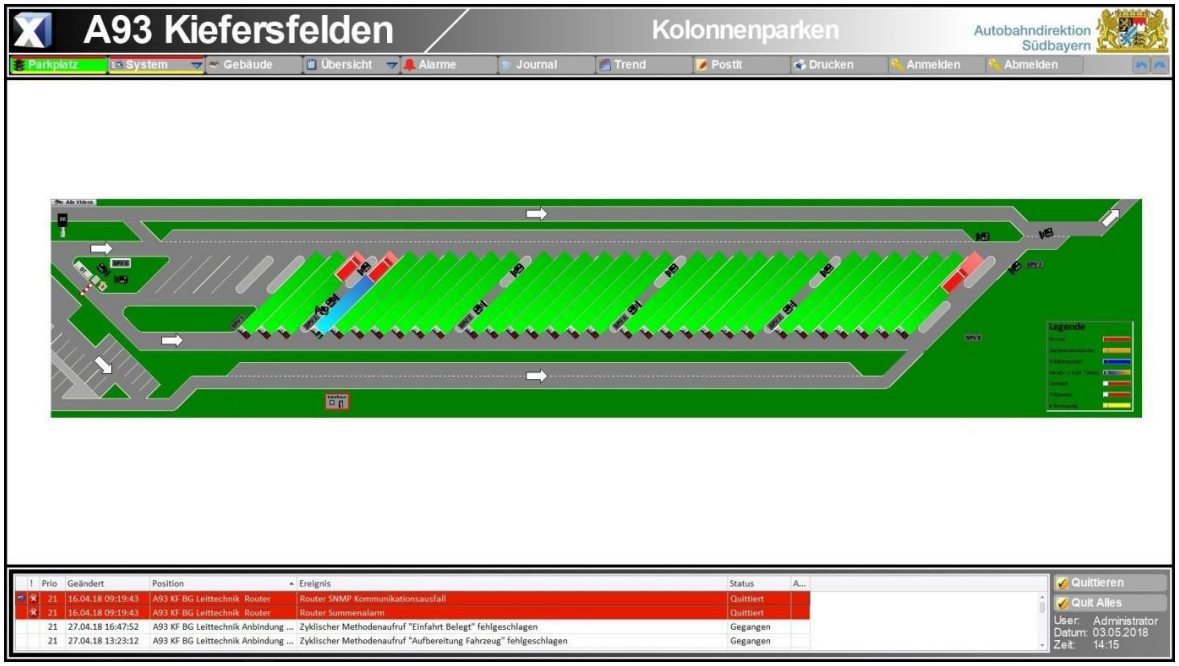
Anlagen, Teil 1

In der nachfolgenden Abbildung wird der schematische Systemaufbau der Anlage dargestellt.



Anlagen, Teil 2

In der nachfolgenden Abbildung wird die Gesamtansicht mit Navigation über zwei Monitore dargestellt.



Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Graz, 20. August 2018

Michael Erwin Hödl